NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

FEASIBILITY COMPARISON AND ANALYSIS OF THE UNIX NETWORK ENVIRONMENT AND THE WINDOWS NT ENVIRONMENT FOR INTEGRATION WITH THE DEFENSE INFORMATION INFRASTRUCTURE (DII)

by

Mark F. Sauer

Timothy J. Smith

John W. Sprague

Joseph E. Staier

September 1996

Thesis Advisor:	Norman Schneidewind
Associate Advisor:	James Emery

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
Feasibility Comparison and Analysis of the UNIX Network Environment and the Windows NT Environment for integration with the Defense Information Infrastructure (DII)

5. FUNDING NUMBERS

6. AUTHOR(S)
Mark F. Sauer, Timothy J. Smith, John W. Sprague, Joseph E. Staier

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

The history of the Department of Defense (DOD) information system technical infrastructure includes a collection of stovepipe, single purpose systems. Recently, the DOD has developed initiatives to help promote the development of common target architectures to which DOD information systems can migrate, evolve, and interoperate. The DOD's Technical Architecture Framework for Information Managers (TAFIM) provides system developers guidance and methodologies for developing standard architectures. The Defense Information Infrastructure (DII) Common Operating Environment (COE) is a development architecture based on the ideas of TAFIM, and provides a framework for designing and building military information systems.

This thesis applies the objectives presented in TAFIM in order to develop an approach for determining which network operating system (NOS) would best facilitate implementations of the DII COE. By first examining the evolution of Navy information systems, and the development of the DII COE, this thesis provides a detailed description of requirements placed on a NOS by a DOD DII COE based information system. These requirements are then used to help understand how TAFIM's objectives apply to NOSs. Two prevalent NOSs, Unix and Windows NT, are evaluated structured on TAFIM's guidance and the requirements of the DII COE. A determination is made based on these guidelines that both NOSs belong in future information systems, for appropriate tasks, based on the DII COE.

14. SUBJECT TERMS Defense Information Infrastructure, Common Operating Environment, Windows NT, Unix, Network Operating System, Technical Architecture Framework for Information Managers, TAFIM, DII, DII COE			15. NUMBER OF PAGES 226 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298

(Rev. 2-89)

Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

FEASIBILITY COMPARISON AND ANALYSIS OF THE UNIX NETWORK ENVIRONMENT AND THE WINDOWS NT ENVIRONMENT FOR INTEGRATION WITH THE DEFENSE INFORMATION INFRASTRUCTURE (DII)

Mark F. Sauer - Lieutenant, United States Navy

B.A., University of Michigan, 1988

Timothy J. Smith - Lieutenant, United States Navy

B.S., United States Naval Academy, 1989

John W. Sprague - Lieutenant, United States Navy

B.S., The Pennsylvania State University, 1990

Joseph E. Staier, Lieutenant JG, United States Coast Guard

B.S., United States Coast Guard Academy, 1992

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1996

Author: _____
Mark F. Sauer

Author: _____
Timothy J. Smith

Author: _____
John W. Sprague

Author: _____
Joseph E. Staier

Approved by: _____
Norman Schneidewind, Thesis Advisor

_____
James Emery, Associate Advisor

_____
Reuben Harris, Chairman
Department of Systems Management

ABSTRACT

The history of the Department of Defense (DOD) information system technical infrastructure includes a collection of stovepipe, single-purpose systems. Recently, the DOD has developed initiatives to help promote the development of common target architectures to which DOD information systems can migrate, evolve, and interoperate. The DOD's Technical Architecture Framework for Information Managers (TAFIM) provides system developers guidance and methodologies for developing standard architectures. The Defense Information Infrastructure (DII) Common Operating Environment (COE) is a development architecture based on the ideas of TAFIM, and provides a framework for designing and building military information systems.

This thesis applies the objectives presented in TAFIM in order to develop an approach for determining which network operating system (NOS) would best facilitate implementations of the DII COE. By first examining the evolution of Navy information systems, and the development of the DII COE, this thesis provides a detailed description of requirements placed on a NOS by an information system based on the DOD DII COE. These requirements are then used to help understand how TAFIM's objectives apply to NOSs. Two prevalent NOSs, Unix and Windows NT, are evaluated according to TAFIM's guidance and the requirements of the DII COE. A determination is made based on these guidelines that both NOSs belong in future information systems, for appropriate tasks, based on the DII COE.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

ACRONYM LIST

3GL	Third Generation Language
ACS	Afloat Correlation System
ADP	Automated Data Processing
API	Application Program Interface
ARPANET	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
ASN (RDA)	Assistant Secretary of the Navy Research Development and Acquisition
ASWTDA	Anti-Submarine Warfare Tactical Decision Aids
ATO	Air Tasking Order
ATP	Advanced Tracking Prototype
BGPHES	Battle Group Passive Horizon Extension System
BSDI	Berkeley Software Design Incorporated
C2	Command and Control
C3I	Command, Control, Communications and Intelligence
C4I	Command, Control Communications, Computers and Intelligence
CCSC	Cryptological Combat Support Console
CCSS	Cryptological Combat Support System
CDE	Common Desktop Environment
CID/DIU	Cryptological Interface Device/Unit
CISC	Complex Instruction Set Computing
CMIP	Communications Management Information Protocol
COE	Common Operating Environment
COTS	Commercial Off the Shelf
CTOS	Convergent Technology Operating System
DBMS	Database Management System
DII	Defense Information Infrastructure
DISA	Defense Information System Agency
DOD	Department of Defense

DON	Department of the Navy
E/IDE	Enhanced Integrated Drive Electronics
EC/EDI	Electronic Commerce/ Electronic Data Interchange
EEI	External Environment Interface
EISA	Enhanced Industry Standard Architecture
EPS	Enterprise Level Parallel Server
EWCM	Electronic Warfare Coordination
FAT	File Allocation Table
FIST/Fulcrum	A Satellite Intelligence and Imagery system
FTP	File Transfer Protocol
GCCS	Global Command and Control System
GCN	Government Computer News
GCSS	Global Command Support System
GOTS	Government Off the Shelf
GUI	Graphic User Interface
HTML	Hypertext Markup Language
I&RTS	Integration and Run-Time Specification
I/O	Input/Output
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronic Engineers
IFS	Installable File System
IRQ	Interrupt Request
ISO	International Standards Organization
JFAC	Joint Forces Air Command
JMCIS	Joint Maritime Command Information System
JOPES	Joint Operations Planning and Execution Service
JOTS	Joint Operational Tactical System
LAN	Local Area Network
LINUX	A Unix variant for PCs named after its developer Linus Torvalds
LOGREQ	Logistic Requirement

MAC	Medium Access Control
MHz	Megahertz
MILNET	Military Network
MRMS	Maintenance Resource Management System
MWM	Motif Windowing Manager
NALCOMIS	Navy Aviation Logistic Command Management Information System
NAVSSI	Navigation Sensor System Interface
NCCS-A	Naval Command and Control system Ashore
NFS	Network File System
NIC	Network Interface Card
NIPS	Naval Intelligence Processing System
NIST	National Institute of Standards and Technology
NITES	NTCS-A Integrated Tactical Environmental Subsystem
NOS	Network Operating System
NSA	National Security Agency
NTCS-A	Naval Tactical Command system Afloat
NTCSS	Naval Tactical Command Support System
NTFS	New Technology File System or NT File System
NWSS	Navy WWMCCS Software Standardization
OBU/OED	Ocean Surveillance Information system (OSIS) Baseline Upgrade
OLE	Object Linking and Embedding
OPNAV	Office of the Chief of Naval Operations
OS	Operating System
OSF	Open Software Foundation
OSI	Open Systems Interconnection
OSS	Operations Support System
PDU	Protocol Data Units
POSIX	Portable Operating System Information Exchange
POST	Prototype Ocean Surveillance Terminal
RAID	Redundant Array of Inexpensive Drives

RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computing
SCSI	Small Computer Systems Interface
SEWC	Space and Electronic Warfare Coordinator
SIPRNET	Secure Internet Protocol Routing Network
SNAP	Shipboard Non-tactical ADP Program
SNMP	Simple Network Management Protocol
SPAWAR	Space and Naval Warfare Systems Command
SQL	Structured Query Language
SQL	Structured Query Language
SSEE	Ship Signal Exploitation
STT	Shore Targeting System
SVGA	Super Video Graphics Array
TAC-4	Tactical Advanced Computer contract 4
TAFIM	Technical Architecture Framework for Information Management
TCP/IP	Transmission Control Protocol/Internet Protocol
TDBM	Tactical Database Management System
TFCC	Tactical Flag Command Center
TIMS	Theater Information Management System
TRM	Technical Reference Model
TSC	Tactical Support Center
U.S.	United States
UB	Unified Build
UIL	User Interface Scripting Language
UPS	Universal Power Supply
VGA	Video Graphics Array
WAN	Wide Area Network
WWMCCS	Worldwide Military Command and Control System

I. INTRODUCTION

A. PROBLEM STATEMENT

The purpose of this thesis is to determine which network operating system (NOS), the Unix NOS or the Windows NT NOS, best meets those objectives outlined in Defense Information System Agency's (DISA) Technical Architecture for Information Management (TAFIM) Technical Reference Model (TRM). The expectation is that this thesis will aid in the selection of a NOS for the Department of Defense (DOD) that best complies with the TAFIM, and best supports the Defense Information Infrastructure Common Operating Environment (DII COE).

There has been an industry-wide trend moving away from independent stovepipe systems to more joint and global information systems. This is often called in the computer industry today a movement towards *open* systems. This trend has been driven largely by the increasing costs of developing and maintaining information systems. Traditionally information systems in organizations were developed with overlapping functionality. Systems were built that performed several common, basic tasks, yet they could not share information. Recently, however, developers have realized the benefits of designing systems based on a set of common building blocks that are readily available to all systems developers. Developers and users benefit from systems built on these concepts because they would promote lower costs, lower investment risks, greater flexibility and greater scalability.

The DOD has also realized the importance of this trend. This thesis will provide, as background, an analysis of the evolution of a DOD information system, and recent efforts by the DOD to achieve a common operating environment for developing information systems. The history of information systems in the DOD underscores the need for the DOD to achieve more open systems. The evolution of the Joint Operation Tactical System (JOTS) demonstrates the DOD's trend towards richer information content, inter-connectivity, and resource sharing through the use of open systems architectures. This need was recognized by the DOD and resulted in the development of several documents, including TAFIM and the DII COE.

The DOD has for years (since the late 1960s) incorporated computer technology to facilitate day-to-day business and information processing. By the late 1980s, DOD information systems were being developed with greater information content, inter-connectivity, and resource sharing. This trend, closely following private sector trends, uncovered a serious flaw in DOD information system development strategies - specifically the development of incompatible stovepipe systems. A major contributing factor for the large number of incompatible systems in the DOD was the multiple operating systems in use.

The DOD has realized, as have other business organizations, the importance in the development of any information system of the selection of a NOS. DOD directives, including TAFIM and DII COE, have incorporated standards that aid in the selection of a NOS for DOD information systems. These documents that detail standards and provide a framework for developing systems have led to the selection of a Unix-based network

operating environment. Recently, however, other operating systems have been introduced in the market place that appear to meet the basic guidelines provided by TAFIM, and in some cases may exceed the capabilities provided by the OSs (Solaris version 2.4 and Hewlett-Packard version 9.0.7) that DOD has selected for its current command system platform, the Global Command and Control System (GCCS).

This thesis provides an analysis of two NOSs, Unix and Windows NT. It evaluates each NOS to determine which overcomes the described shortcomings of previously developed DOD information systems (the shortcomings that were later described in TAFIM as the objectives to be used for the development of future DOD information systems).

B. MOTIVATION

The authors feel that the DOD has maintained its historical ties to the Unix operating system when developing current information systems such as the Joint Maritime Command Information System (JMCIS) and the GCCS. While Unix may have been the operating system of choice in the past, current market trends and technologies - specifically, those found in Windows NT - may have qualities that should be evaluated and compared to Unix. The TAFIM TRM criteria and objectives of TAFIM should reflect the fast changing and rapidly advancing trends of the market place.

C. BACKGROUND

The DOD has become increasingly aware that systems integration is essential in cost cutting and downsizing, as well as from a simple systems management point of view. In the following sections, we discuss each component of the evolution of the current DOD information system. Each system is incorporated into the next system, and eventually becomes a new piece of the picture that results in the DOD's current information system. The background concludes with a description of TAFIM and why the TRM section of TAFIM is used as the "hallmark" comparison that the two NOSs are measured against.

1. JOTS & NTCS-A

The JOTS command and control system began as a Command, Control, Communication, and Intelligence (C3I) prototyping effort in 1986. Since 1986, variants of the JOTS system have been installed on deploying ships and shore stations to aid in the command, control, and track management of units at sea (friendly, unknown, and hostile). Advancements in computer technologies have led to the development of a JOTS-derivative system called the Navy Tactical Command System, Afloat or NTCS-A. Figure 1 shows the evolution of the JOTS system to the NTCS-A system.

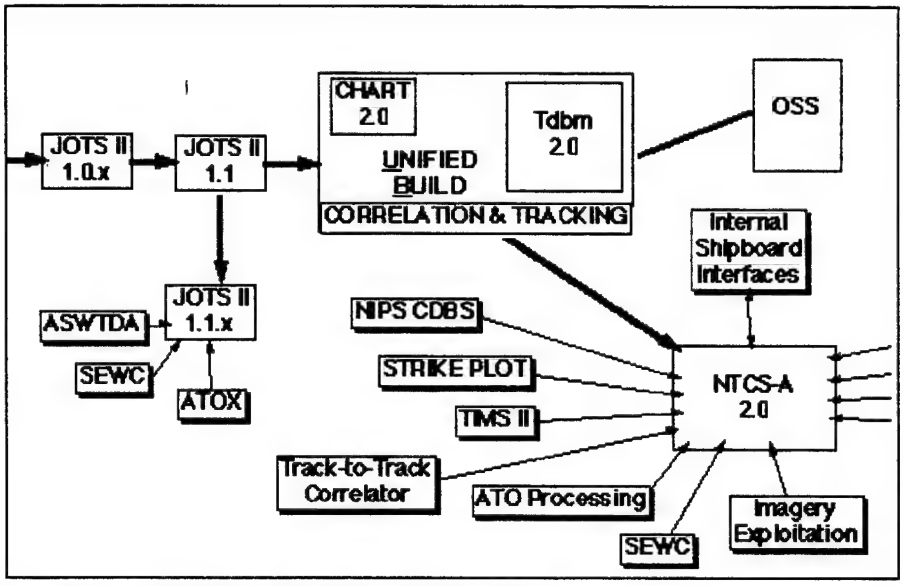

Figure 1. NTCS-A Evolution [NRAD02]

Since the initial prototyping efforts in 1986, versions of one or both systems have been installed onboard over 200 U.S. Navy ships, at several U.S. Navy ashore intelligence centers, onboard U.S. Coast Guard cutters, onboard allied ships, and at various allied sites. These systems clearly demonstrated their value as key Command and Control (C2) systems for the United States and its NATO Allies during the Persian Gulf War.

As JOTS matured further and as other C3I systems were developed and deployed, it became apparent that there was much duplication of software and functionality across systems, and that this duplication led to increased development, maintenance, and training costs. Interoperability was practically nonexistent across systems even when systems followed the same set of standards. Perhaps the most serious impact, however, was that operators were often given conflicting information from multiple systems even when the systems were presented with identical data.

Based upon this observation and experience, the Space and Naval Warfare Systems Command (SPAWAR) directed that the afloat software be abstracted into a common "core" set of software that could be used throughout the afloat community as a basis for all afloat community systems. This effort led to a set of common software called Government Off-The-Shelf (GOTS) version 1.1. SPAWAR then directed that this approach be extended to include not only the afloat community, but the ashore community as well. This way both communities could share the same common set of software to reduce development costs, ensure interoperability, and reduce training costs. This effort resulted in a collection of software commonly referred to as the Unified Build (UB) version 2.0 and also referred to as GOTS 2.0 [NRAD03].

This software is now deployed both afloat, in NTCS-A, and ashore, in a system referred to as Operations Support System (OSS) or Navy Command and Control System - Ashore (NCCS-A). The strength of these two systems is that they are built on top of a common set of functions so that advancements and improvements in one system are immediately translatable to advancements and improvements in the other system. The UB software is presently the basis for numerous other efforts, including systems for the Navy, Marine Corps, Coast Guard, and, increasingly, for the joint community. The programmatic strategy is to integrate all these software programs into a common infrastructure called JMCIS. Figure 2 is a description of how NTCS-A and OSS integrate into JMCIS.

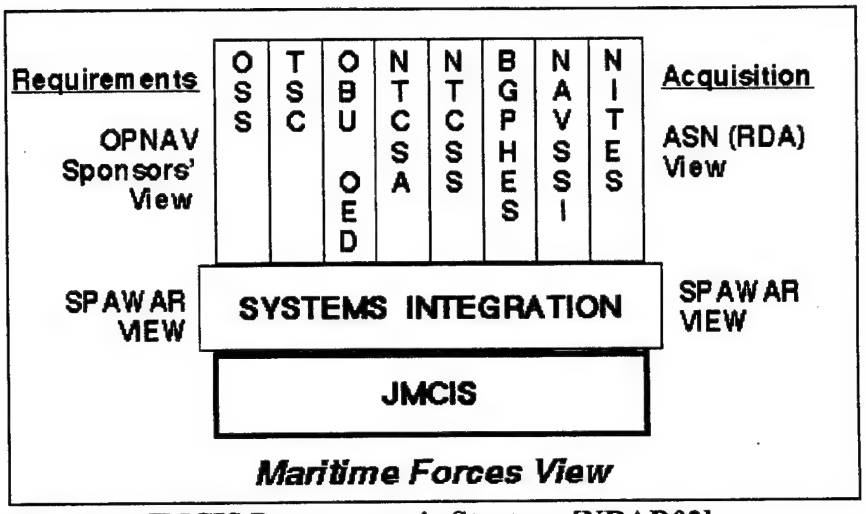

Figure 2. JMCIS Programmatic Strategy [NRAD02]

2. JMCIS

JMCIS was the next step in the evolution of the Navy's information system. JMCIS is both a development concept and an information system currently deployed to the fleet. To properly understand what JMCIS is, it is important to consider the viewpoint of the end user (sailor/soldier), the military program manager, and the system developer. To the end user, JMCIS represents a command information system that is distributed across a local area network (LAN) of workstations. An operator is able to access all required functionality from any workstation, regardless of where the workstation is located or where the actual processing takes place on the LAN. Functionality that exists throughout the system, but which is not useful for the operator's tasks, is hidden so as to not overwhelm and confuse the operator with extraneous features. An operator with a different set of tasks, however, may see a different set of functionalities, but both operators will perceive that the system looks and operates in the same manner. Moreover, to both operators, JMCIS will appear to be the same command information system in use

by other U.S. military services. This is increasingly important in the joint community where joint exercises, such as Joint Forces Air Command (JFAC), are performed to reassign command responsibilities from one service to another. While the end user sees JMCIS as an information system on a LAN, it is actually only part of the complete system that exists across a much larger wide area network (WAN).

From the perspective of a military program manager, JMCIS presents the opportunity to create an umbrella program, that encompasses several aspects of the program manager's problem domain. In a downsizing military, the program managers need to perform tasks with increased efficiency and less resources. For example, in the late 1980s, SPAWAR PD-60 (Navy-Afloat program management) supported the JOTS program for battlegroup track database management, NIPS (Naval Intelligence Processing System) for database management, TIMS (Tactical Information Management System) for automatic display of status information, FIST/FULCRUM for imagery acquisition, and a host of other related programs just to support the battlegroup commanders. Ashore program managers had similar programs to support Navy intelligence centers. JMCIS now provides Navy program management with an umbrella program which combines the requirements into a single, consolidated, coordinated system. The associated cost savings are substantial. Figure 3 shows the incorporation of the many stovepipe programs into JMCIS.

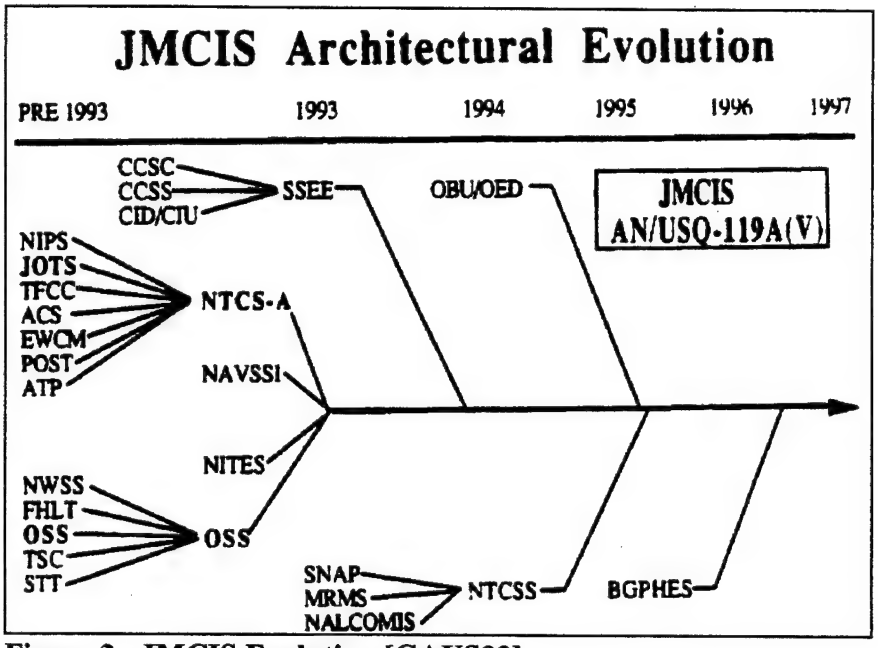

Figure 3. JMCIS Evolution [GAUS93]

From the perspective of a system developer, JMCIS is an *open* architecture and a software development environment that offers a collection of services and already built modules for command information system components. The system developer's task is to assemble and customize the existing components from JMCIS while developing only those unique components that are peculiar to his particular mission requirements. In many, if not most, cases this amounts to adding new "pull-down menu entries."

In many ways, the JMCIS model is similar to the Microsoft Windows paradigm. The idea is to provide a standard environment, a set of standard off-the-shelf components, and a set of programming standards that describe how to add new functionality to the environment. JMCIS is also a superset collection of software. More precisely, JMCIS should be viewed as a collection of several related items required for any command information system development. JMCIS is all of the following:

1. A clearly defined set of functions (modules) that constitute a command information system. These functions, along with the software that implements them, form the JMCIS core, which includes track management, correlation, communications, and tactical display components.

2. A precisely defined architecture for how the modules will interact and fit together, and a definition of the system level interface for the modules.

3. A standard operating environment that includes "look and feel," operating system, and windowing environment standards.

4. A commercial set of Unix, X-Windows, and Motif standards and software.

5. A collection of already developed and tested modules that implement the above functions according to the architecture described above, and a set of Application Programmer Interfaces (APIs) for accessing these functions.

6. A collection of already developed and tested Tactical Decision Aids (TDAs) and support functions (range and bearing calculations, Closest Point of Approach (CPA), etc.) that may be incorporated into a command information system.

In addition to understanding what JMCIS is, it is important to understand what it is not. JMCIS is none of the following:

1. The same as UB, NTCS-A, or OSS. Each of these efforts have contributed software to the JMCIS superset. UB is now multiple JMCIS segments while NTCS-A and OSS are replaced by appropriate JMCIS variants.

2. A solution for all command and control problems. However, a large number of applications, whether or not related to Command and Control, share common requirements with JMCIS.

3. Government modified COTS products. Commercial software is used whenever practical, but the executable code and data files are not modified except to customize the COTS products as described in the vendor COTS documentation. For example, the Unix operating system used within JMCIS is fully Portable Operating System Information Exchange (POSIX) compliant but is configured to meet requirements (shared memory, message pool, etc.) using vendor-supplied techniques. This is equivalent, in the personal computer (PC) arena, to editing the disk operating system (DOS) CONFIG.SYS and AUTOEXEC.BAT files. Virtually all PC software products require customization of these files.

4. A deviation from accepted industry standards. Commercially available standards, such as the Motif Style Guide, are used to the fullest extent possible with customizations made within the scope of the standards to allow for the uniqueness of the military environment. For example, military systems must accommodate low light/red light/blue light operating conditions.

5. Vendor proprietary (by some definitions). Vendor proprietary products are used (such as Unix, Oracle, and Sybase) but these are vendor proprietary implementations of industry standards. No SPAWAR funded JMCIS software is vendor proprietary. [NRAD01]

3. GCCS

GCCS is a relatively new information system which has recently been deployed at several operational Commanders in Chief units (CINCs).[1] GCCS was developed to improve the joint warfighter's ability to effective execute a number of missions, while at the same time integrating Command, Control, Communications, Computers and Intelligence (C4I) systems from services and DOD agencies. GCCS is intended to support the joint warfighter in operations ranging from peacetime humanitarian operations to non-nuclear strategic war. GCCS was developed for the DOD by DISA. The initial objectives of GCCS was the replacement of the World-Wide Military Command and Control System (WWMCCS) and the implementation of the C4I for the Warrior Concept.[2] The goal is to have GCCS become the "single, global command, control, communications, computer and

---

[1] Global Command Support System (GCSS) is presently under development and is targeted for the warfighting support functions (logistics, transportation, etc.) to provide a system that is fully interoperable with the warfighter C4I system. When implemented to its fullest potential, GCSS will provide both warfighter support and cross-functional integration on a single workstation platform.

[2] C4I for the Warrior provides guidance for DOD information managers to solve the interoperability issues throughout the services. It is designed to support the joint warrior on the battlefield, allowing timely, accurate, and relevant information to be "pulled" by the warrior.

intelligence system to support the warfighter, whether from a foxhole or from a command post." [DISA06, p. ii]

The target GCCS architecture is based on the JMCIS architecture, the Army AWIS, and advanced information processing and communications technologies. It supports a wide variety of command and control missions and functions. GCCS version 1.1 included mission applications from a variety of other programs operating in a "federated" mode. Like the JMCIS architecture, GCCS was developed with the idea of trying to improve on architectures of previous DOD information systems. Other considerations stated in the GCCS COE during development were the rapid changes in technological areas and the changing national strategy. The result was to develop an evolutionary migration strategy that provided for the following:

- Keeping the war fighter involved at all levels.
- Allowing the war fighter to retrieve, manipulate, share, and view database information as needs change.
- Providing a unifying architecture that provides a path for migration.
- Building an open infrastructure flexible enough to easily accommodate future requirements.
- Relying upon the military services and agencies of GCCS components, data sources, and information sources.
- Allowing a vehicle for technology insertion [DISA06, p. 1-1].

Conceptually, GCCS consolidates and modernizes all existing command and control functions in the DOD. DISA provides systems engineering services for designing future information systems [FONG94]. The GCCS architecture provides integrated

information processing and transport capabilities in support of a variety of warfighting functions and missions. In fact, GCCS runs the JMCIS software segment, as well as other service specific software applications. GCCS requires reliable high-speed networking, multi-media conferencing, distributed simulation, and training. In addition, GCCS accommodates many information applications through shared processing, distributed data, multimedia communications, and centralized monitoring and control. Functionally GCCS takes JMCIS one step further, by integrating all of the information system needs for tactical support into one standardized system networked across a LAN/WAN.

The desire for DISA is to have GCCS migrate to full compliance with TAFIM. The GCCS COE adopts all of the objectives for developing information systems stated in the TAFIM TRM. When achieved, it is thought that GCCS will benefit from having an open system architecture. GCCS is intended to be hardware independent, and capable of operating on a range of open systems platforms. Its operating system is to be a standards-based operating system. At present, only two variants of the Unix NOS have been implemented in GCCS, HP-UX v9.01 and Sun Solaris v2.3.

4. Defense Information Infrastructure (DII)

DII was created during the early stages of GCCS development because DISA realized that instead of developing a specific intermediate system, a more all-encompassing open-system concept was needed. This new COE would allow all new systems, applications, and databases to be developed in an open-system environment ensuring a reduction in common problems such as data redundancy/duplication, and system security.

DII is a design intended to permit all DOD information systems to communicate and share information through the use of standardization. DII achieves this standardization by defining several "open architectures" that are platform independent. At the base of this standardization effort is the definition of a Common Operating Environment (COE). The COE concept is best described as:

- An architecture that is fully compliant with TAFIM Volume 3
- An approach for building interoperable systems
- A collection of reusable software components
- A software infrastructure for supporting mission area applications
- And a set of guidelines and standards

The guidelines and standards specify how to reuse existing software, and how to properly build new software so that integration is seamless and, to a large extent, automated [DISA02]. GCCS and GCSS presently use the DII COE. Both use the same infrastructure and integration approach, and the same COE components for functions that are common between the two systems. The DII COE, being the current information system thrust, will be discussed in greater detail in the following chapter.

5. Technical Architecture Framework for Information Managers (TAFIM)

DISA began in 1993 to publish a series of documents describing systems development guidelines for the DOD. These documents comprise the DOD TAFIM. TAFIM is intended to "provide guidance for the evolution of DOD technical

infrastructure" [DISA03, Vol. 1 p. 3]. TAFIM evolved from the many lessons that the DOD had learned in the development of information systems like JOTS and JMCIS. It was a giant step by the DOD to set down on paper requirements and standards for systems developers to adhere to when designing future DOD information systems. The following describes TAFIM's background and intent.

An information system includes support and mission-oriented applications, computing platforms, and communications networks. The current DOD information system technical infrastructure consists largely of stovepiped, single-purpose, and inflexible systems that are costly to maintain. These systems reflect a multiplicity of approaches to migrate toward open systems with each one progressing on its own path with limited attention to interoperability.

The evolving DOD enterprise vision for information management emphasizes integration, interoperability, flexibility, and efficiency through the development of a common, multi-purpose, standards-based technical infrastructure. This vision requires a new paradigm for building technical architectures and information systems that improve the effectiveness of functional operations to include their efficiency and use of technology throughout the DOD.

The emerging concepts for warfighting depend upon information being managed as a department-wide resource. Joint campaigns should fully exploit the "information differential," which is the superior access to and ability to effectively employ information on the strategic, operational, and tactical level that advanced U.S. technologies can provide to our forces. This information differential requires a smooth seamless interface

between the "foxhole" and the support base, between intelligence and operations, and between the DOD and its suppliers. However, before today there has been no unifying DOD information management technical architecture guidance that can satisfy these goals.

In the absence of DOD-wide guidance, the services, agencies, and CINCs have independently developed a wide range of architectures to manage and control their technical infrastructures. Reference models, information architectures, communications architectures, mission architectures, and various other architectures are now used to manage the design and development of technical infrastructures and information systems within the services, agencies, and CINCs.

The TRM for information management was the initial effort to bring commonality and standardization to the technical infrastructure. The TRM addresses the services and standards needed to implement a common technical infrastructure. A single technical architecture framework was needed to integrate these efforts and drive systems design, acquisition, and reuse throughout the DOD.

The single technical architecture framework is the TAFIM. It provides the DOD-wide framework to manage multiple technical architecture initiatives. It is intended to achieve the following results:

- The use of common principles, assumptions, and terminology in the DOD Component (services, agencies, and CINCs) technical architectures
- The definition of a single structure for the DOD technical infrastructure components (system components) and how they are managed
- The development of information systems in accordance with common principles to permit DOD-wide integration and interoperability

TAFIM provides guidance for the evolution of the DOD technical infrastructure; it does NOT provide a specific system architecture. Rather, it provides the services, standards, design concepts, components, and configurations that can be used to guide the development of technical architectures that meet specific mission requirements.

TAFIM is independent of mission-specific applications and their associated data. It introduces and promotes interoperability, portability, and scalability of DOD information systems. TAFIM is an enterprise-level (departmental or DOD level) guide for developing technical architectures that satisfy specific functional requirements. It also provides an organizational-level guide and link to the enterprise level. To achieve an integrated enterprise, it is assumed that all information systems must interoperate at some time. Therefore, their architects and designers should use TAFIM as the basis for developing a common target architecture to which systems can migrate, evolve, and inter-operate. Over time, interoperability between and among the number of systems will increase, providing users with improved services needed to achieve common functional objectives. To achieve portability, standard interfaces will be developed and implemented. Scalability will be developed in mission applications to accommodate flexibility in the functionality. Proper application of TAFIM guidance can:

- Promote integration, interoperability, modularity, and flexibility
- Guide acquisition and reuse
- Speed delivery of information technology and lower its costs

TAFIM applies to information system technical architectures at all DOD organizational levels and environments (e.g., tactical, strategic, sustaining base, interfaces

to weapons systems). TAFIM is mandatory for use in DOD [PAIG95]. The specific technical architectures for missions and functions will be developed using standard architecture guidance and development methodologies provided by the TAFIM.

D. SCOPE & ORGANIZATION

The scope of this thesis includes an analysis of the Unix and Windows NT network environment as they apply to the TRM. Based on this evaluation, the thesis provides a recommendation of the optimal network environment to provide a standardized foundation for the DII COE is provided.

In order to compare Unix and Windows NT, it is imperative to analyze the DII COE requirements. With this understanding, it is necessary to pick some criteria and evaluate the relative performance of each operating environment. The authors feel that an appropriate evaluation method is one that returns to the objectives stated in TAFIM. TAFIM has eight objectives, which the authors chose to use as criteria when evaluating the two network operating environments. Although there are numerous criteria that could have been used, the authors feel that these eight criteria best summarize the objectives that the DOD originally subscribed to as guidance for the evolution of the DOD technical infrastructure. These objectives originated in TAFIM from lessons learned from the development and failures of past DOD information systems. The eight objectives are:

1. Improve User Productivity
2. Improve Development Efficiency
3. Improve Portability and Scalability

4. Improve Interoperability
5. Promote Vendor Independence
6. Reduce Life Cycle Costs
7. Improve Security
8. Improve Manageability

Each of these criteria and their tradeoffs are discussed later in this thesis. Each section of Chapter III describes a criterion as defined by TAFIM, discusses the authors' interpretation of the criterion, discusses how Unix and Windows NT meet or do not meet the criterion, and finally states which NOS best suits the DII COE. A conclusion will follow with a summary of findings, solutions, and recommendations, as well as suggested further studies.

II. DEFENSE INFORMATION INFRASTRUCTURE COMMON OPERATING ENVIRONMENT (DII COE)

A. ANALYSIS

As described in Chapter I, the DII COE architecture was designed with the intent of providing an innovative framework for designing and building military systems. The resulting COE is "very simple and straightforward, but powerful in its ability to tailor a system to meet individual site and operator requirements." [DISA04, p. 2-1] Great importance was given in designing flexibility in the DII COE. This flexibility is needed as different systems are designed to meet changing and diverse military needs. This flexibility is also needed in selecting COE approved NOSs. The NOS is often considered the brain of the information system; it facilitates communication and resource sharing throughout the system. It provides the framework for the system. An understanding of the potential environment is also needed when determining a NOS, because the environment determines requirements for the NOS.

1. Environment

DII is intended to provide real time information management capabilities to all mission areas of the DOD. It is intended to provide the warfighter with information capabilities to achieve success. The role of the U.S. armed forces in supporting U.S. interests is defined in the National Military Strategy (NMS). This chapter helps outline the potential missions, tasks, and capabilities of the armed forces of the U.S. in which the DII COE is important.

The NMS continues to refine two fundamental strategic concepts: overseas presence and power projection. Both of these concepts are required to support our national interests. These strategic concepts are achieved through accomplishing three sets of tasks: 1) peacetime engagement, 2) deterrence and conflict aggression, and 3) fighting and winning wars.

The NMS clearly states that the military's primary responsibility is to fight and win wars. This goal is achievable by following certain principles, including use of decisive force, wartime power projection, having clear objectives, fighting joint wars, countering weapons of mass destruction, and winning the information war. Uncharacteristic of previous NMSs is the inclusion of winning the information war as a principal of winning wars. The NMS states that emphasis must be put on the collection, processing, and transmission of intelligence data. "The services and combatant commands require such fused information systems. These systems enhance our ability to dominate warfare." [NMSG95, p. 15]

The military, as evidenced by the NMS, is realizing the importance of information as a resource. Uncertainty in the world and increasingly changing roles for the military has necessitated this need. Today, the military can find itself executing numerous missions, including humanitarian operations, peacekeeping, nation assistance, counter-drug operations, and counter-terrorism operations. So many diverse missions make the military more reliant on communications and information services. Effectively using these resources will improve the military's performance. The need is to define an infrastructure to accomplish this.

DII COE is a framework designed with these needs in mind. As can be seen in Figure 4, GCCS is one of many systems that implements the DII COE. It demonstrates some of the wide ranging capabilities that the DII COE provides. GCCS is a command and control system designed to support the warfighter. Physically, it consists of a number of workstations distributed across a classified network. Communication mediums allow warfighters across the network to share information, and plan, perform, and collaborate on missions. With GCCS, warfighters are able to more effectively plan and collaborate on functions such as force deployments, complex multi-force air tasking orders, intelligence analysis, and maintaining current displays of force deployments, both joint and enemy.

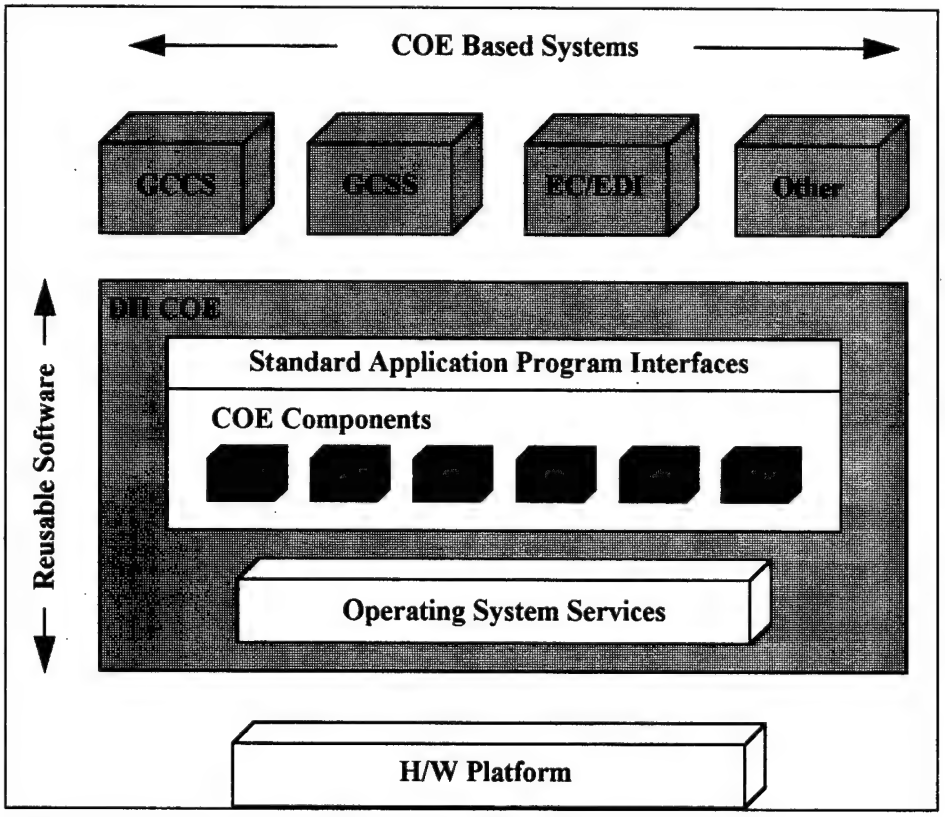

Figure 4. GCCS as a portion of a COE based system

The GCCS system provides a suite of capabilities across a number of mission application areas that include the following [DISA04]:

- Manpower Requirements Analysis
- Force Planning
- Collaborative Mission Planning
- All Source Data Fusion & Correlation
- Office Automation
- Logistics Support
- Status of Readiness Reports
- Cartographic and Imagery
- Display and Analysis
- Transportation Planning
- Resource Management
- Fuel Resource Planning
- Teleconferencing
- Scheduling and Movement
- Medical Planning
- Comms and Msg Handling
- Intelligence Analysis

Additionally, the DII COE must support other Information Systems (ISs) such as GCSS and EC/EDI.

With a system capable of providing such diverse capabilities, the potential exists for operators to experience information overload. Operators in such an environment could also access information that they might not have either clearance for or a need to know. To alleviate these potential problems, DII COE designers provided system administrators with the ability to install only those portions, or segments, of a system that a command and its operators need to perform their mission-related tasks. The system administrator customizes features of the system for each user. This powerful feature allows the capabilities of the system to be tied to the user, not the workstation. This is very important in the shared workstation environment that DII COE based systems will operate in. In a system such as GCCS, a yeoman can access his workstation for office automation, while his operations department head can access classified Status of Resource and Training Reports (SORTS) on the same machine. Each user can access only the information that he needs to perform his mission.

The environment calls for the DII COE framework to be capable of providing for a variety of missions. Not only will it serve an environment of diverse missions, but it will be accessed by users of dramatically different needs. These characteristics put constraints on the COE design, as well as necessitating certain needs on what operating system it will employ.

2. Common Operating Environment Design

The need to provide so many capabilities in DOD information systems necessitated the need for a COE. The power of the DII COE, as stated in the DII COE Integration and Runtime Specification (I&RTS), is its ability to tailor a system to both an individual site and an operator's requirements. Design of the COE was driven by principles, not applications. Selection of the actual components for a particular instance of a COE determine the functionality of that system. The GCCS is a COE-based C4I system.

The basic building block of the DII COE is called a segment. Segments are defined as individual self-contained software packages that provide different functionality. They include all software except the operating system. Both the infrastructure and mission application software are developed as segments. This allows the ability for individual sites to tailor the system to their needs, by enabling administrators to load only needed segments.

Figure 5 is a diagram that shows the current DII COE. The figure depicts the relationship between the COE, component segments, and application segments. The COE contains building blocks such as the operating system, security services, communication services, and a windowing system. Mission-specific application segments "plug into" the COE via standard application program interfaces. The DII COE states the analogy that

the COE component segments are similar to built-in features, whereas mission application segments are added to the COE; it is like adding additional circuit boards to a mother board. [DISA04, p. 2-4]

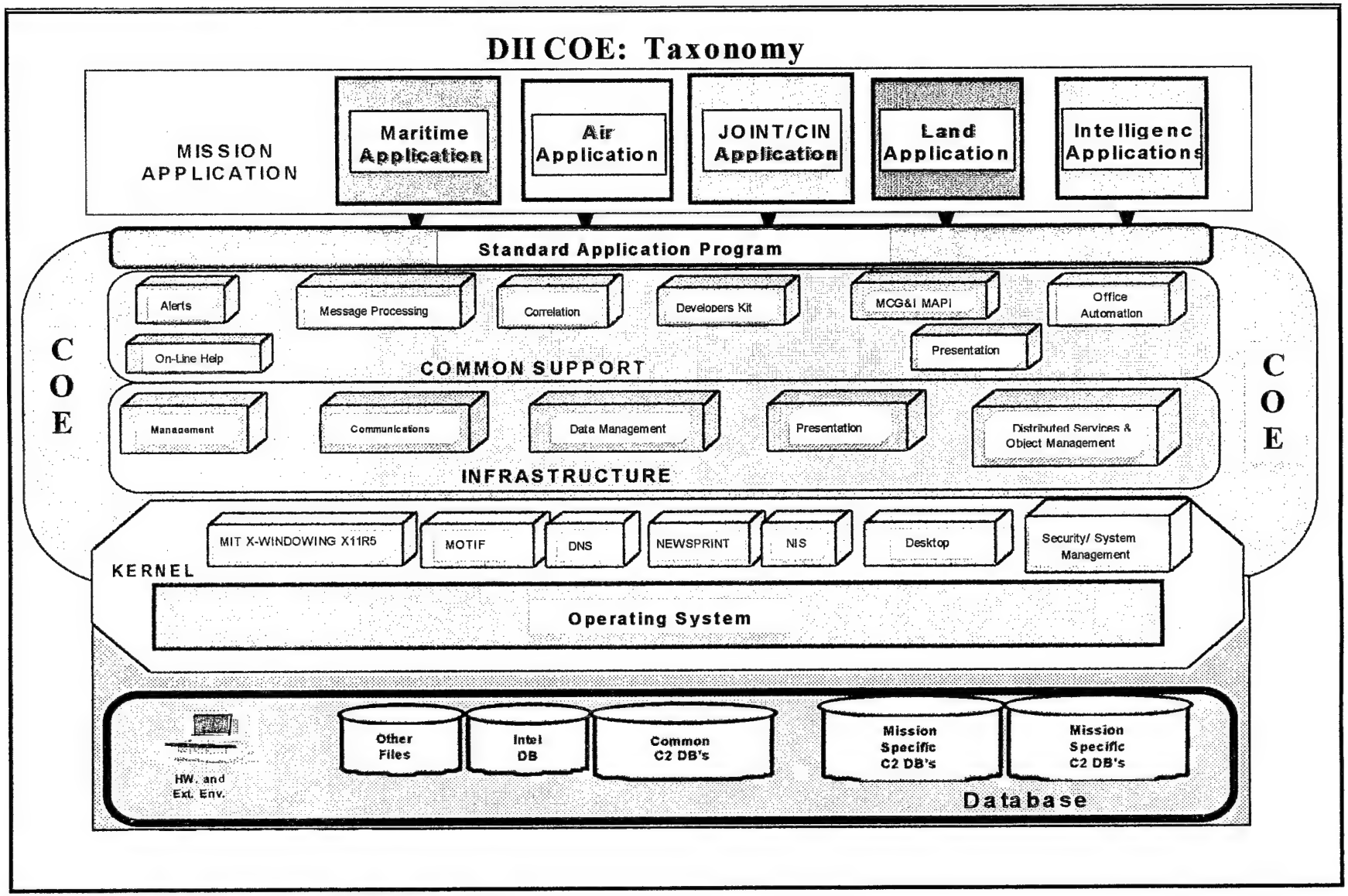

Figure 5. DII COE Taxonomy [BUTL96, p. 8]

The API[3] layer in the COE defines how segments may plug into the COE. APIs are the only way for segments to access services provided by the COE. Therefore, all software, GOTS or COTS, must contain standard APIs to access the COE.

Due to the complex environment discussed in the previous section, DII COE designers originally created 19 different functional areas. This proved to be unmanageable and communications between different workgroups became infeasible. Because of this unmanageability, designers then decided to approach the DII COE from an architectural perspective rather than a functional perspective. As a result, a collection of Common Support Applications and a collection of Infrastructure Services were developed along with some new logistics services. This new architecture is shown below in Figure 6. This architectural approach greatly reduces the communications burden in and between working groups. This figure does not show new logistics support services nor the financial services (e.g., EC/EDI).[4] [DISA04, p. 2-17]

---

[3]An Application Programmer Interface (API) is a programmer's guide that describes the COE software libraries and services, and how to write software modules that interface with and use the COE services. [DISA04, Glossary]

[4] For more information on the COE taxonomy, see the Architectural Design Document for the Global Command and Control System (GCCS) Common Operating Environment (COE).

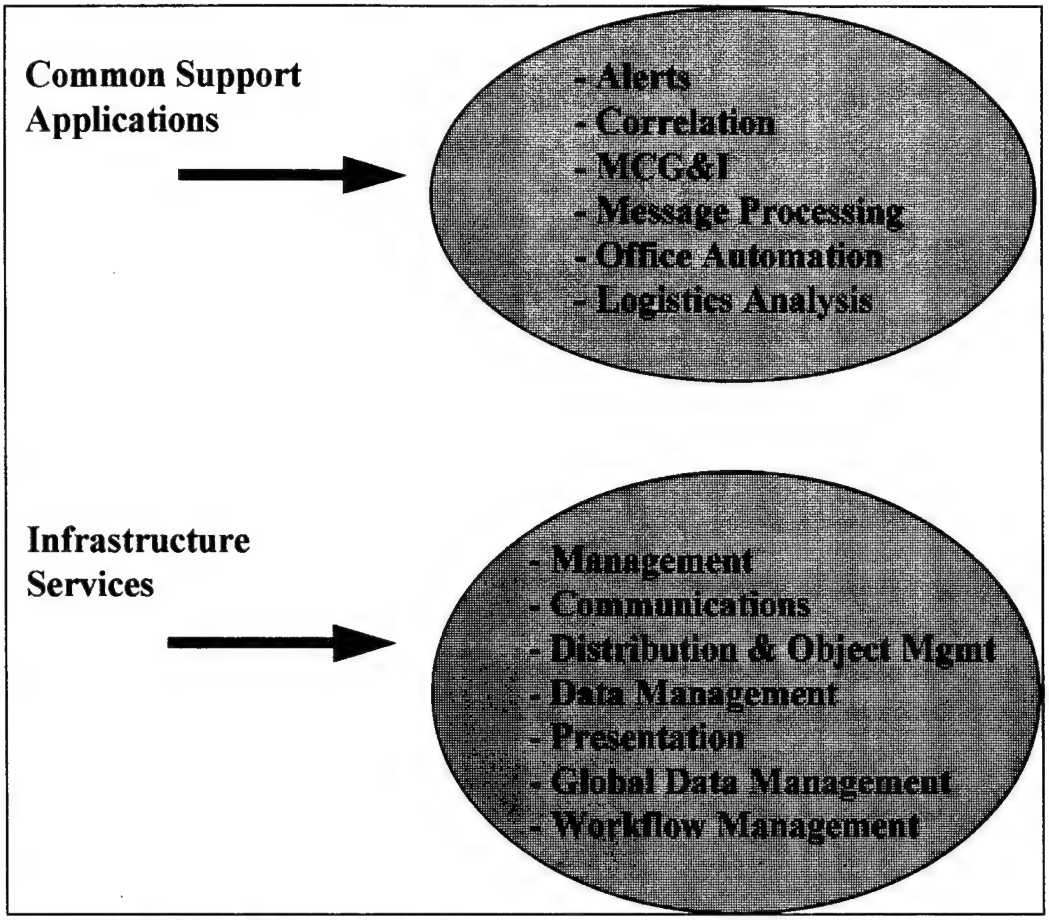

Figure 6. DII COE Architecture

The primary focus of the Infrastructure Services is to provide a framework in which the flow of data throughout the system can be distributed and managed. The subsections under Infrastructure Services and a description include:

1. Management: The management of the data includes system security, system administration, and network maintenance.
2. Communications: The communications services provided a means of data exchange with external systems.
3. Distribution and Object Management: The distribution and object management allows true distributed processing in a client server environment.
4. Data Management: The data management portion includes relational database management as well as file management over a distributed system.

5. Presentation: Presentation services is the part of infrastructure services which controls the interaction with humans through a Graphic User Interface (GUI) interface.

6. Workflow and Global Data Management: Workflow and global data management is the section that works toward the management of logistical data (e.g. LOGREQs, parts inventory, etc.)[DISA04, p. 2-18]

Infrastructure Services originated from the C4I problem domain, but the services provided are largely independent of any particular application.

On the other hand, Common Support Applications are normally defined in a particular problem domain. The Common Support Application area includes:

1. Alert Services: Alert services are responsible for managing alert messages throughout the system, ranging from system administration messages like paper jams, to mission oriented messages like incoming missile messages.

2. Correlation Services: These services provide a consistent view of the battle space by correlating various informational inputs.

3. MCG&I: MCG&I services are responsible for displaying maps and other images from diverse sources.

4. Message Processing Services: These services are responsible for the distribution of military formatted messages.

5. Office Automation Services: Office automation services handle the typical office processes such as word processing, spreadsheets, and electronic mail.

6. Logistics Analysis: Logistic analysis includes common functions for analyzing and viewing logistical information. [DISA04, p. 2-18]

The selection of the software components which meet the responsibilities outlined in the DII COE has not been completed. In fact, it will be an ever-changing process as needs change and new software is developed which better meets the needs of the DOD. In order to be able to continue the process of software improvement, the COE will "... preserve backwards compatibility so that mission applications are not abandoned just because there is an update of the COE" [DISA04, p. 2-18]. This relatively new thought

process emphasizes "... incremental development and fielding to reduce the time required to put new functionality into the hands of the warrior, while not sacrificing quality nor incurring unreasonable program risk or cost." [DISA04, p. ii]

The COE implementation strategy has been designed so that the DOD can migrate easily to new systems over time. This capability is commonly referred to as scalability. In order to facilitate future scalability, the DII COE uses the Institute of Electrical and Electronic Engineers (IEEE) POSIX P1003.0 open systems standards. POSIX is a set of standards that define the interfaces between applications and the operating system. It is important to note that POSIX does not define the actual implementation of the operating systems. Portability and interoperability of applications are the critical issues, not the code inside the operating system. [SING95, p. 2] POSIX defines a set of Application Program Interfaces (APIs), and is discussed later in this chapter.

In order to further protect future scalability and the DII COE's ability to migrate to new systems, the DII COE defines "public" and "private" APIs. Public APIs are APIs that will continue to be utilized over the life of the COE. All new components should be developed using these public APIs. The private APIs are transitional APIs to allow legacy components to continue to work until the need for it has passed or it has been replaced with a more current version.

Figure 5 shows the current taxonomy and cross-section of the entire DII COE. It is a layer of abstraction independent of the hardware, NOS, and mission application software which contains a set of Common Support Applications and Infrastructure Services. It also is the layer in which the APIs reside. The DII COE is not a vertical slice

such as Intuit's Quicken financial software running on a Windows family OS or NOS using an Intel machine architecture.

B. OPERATING SYSTEM REQUIREMENTS IN THE DII COE ENVIRONMENT

1. Operating System Requirements

The operating system has requirements placed on it due to the environment (as described above) in which it operates. The design of the DII COE also places further requirement on the NOS. Several of these characteristics are common features desired in network environments today, but require more attention in the complex, multi-faceted environment that the DII COE is designed for. These NOS characteristics include: real time capabilities, multi-user capabilities, common user interfaces, specific security requirements, and reliability requirements.

If a system is going to be used to gather data and present a tactical information display, it must be a real-time system or near-real time[5] (i.e., it must guarantee that certain functions will happen within exact time constraints). A NOS must be chosen with this in mind. NOSs based on Netware, for instance, cannot do this, and therefore cannot be used for real-time systems [GASK95, p. 1086].

---

[5] Whether a system is real-time or not depends on the data being delivered. A JOTS screen might be considered real-time if it is updated every second while a fire-control system might need new information every 1/100th of a second to be real-time. A tactical support real-time type system is our baseline in this discussion.

This real-time tasking can be accomplished in many ways. Some variants of Unix, as well as Windows NT, for instance, do this by what is called preemptive scheduling or preemptive multitasking. Preemptive scheduling is a sophisticated way to describe how a system goes about setting priorities. This allows the system to schedule jobs to begin at set time intervals or at set conditions, no matter what other set of system functions are active. [FEIB95, p. 616] DII COE implementations will require this type of capability.

Another need for the DII COE NOS is for multi-user capabilities. Many different people on board a ship might need access to the same database at the same time. The Combat Information Center (CIC) and the bridge might both need the information contained in the JOTS database at the same time. Also, the supply officer might be running an inventory check at the same time an order is being placed for a new part. These examples all potentially exist in a typical DII COE implementation and necessitate the need for the NOS to be able to support more than one user at a time. A distributed database is also a direct derivative of multiple operators using the same databases from different locations.

Another multi-user issue encountered on board ship is the simple fact that every user does not have his own personal computer. Logins and passwords must be present so that users are allowed access only to information that should be available to them (e.g., e-mail accounts, etc.).

This requirement for multi-user availability necessitates the need for the system to have certain security requirements. DII COE systems are designed with the idea that there will be multiple users with multiple needs. These users should only be allowed to access information that they need to complete their tasks. The NOS needs to be able to support

some type of access control. Other security features would also be needed, including audit trails.

A common set of user interfaces (UI) is also a requirement of the NOS. With the ever changing personnel in the ship's company and the possibility personnel rotating between watchstations, a common UI is essential. On-the-job training is the norm, especially in the surface community, so intuitive UIs are essential to reduce lost productivity. If all the UIs on deploying units and shore commands are the same, or at least very similar, then productivity is increased.

Availability and reliability are important in almost all environments, but especially in one where a multitude of programs are running at the same time on the network. NOSs must be able to be relied on at all times for mission-critical information. A supply program crashing while communicating with a remote server should not interfere with a tactical support program's operation.

Another important function that is required of the NOS is ease of network maintenance. Networks are becoming increasingly complex and the DOD's reliance on them is increasing everyday. Currently, however, the DON does not have billets on ships (and few, if any, on small shore units) to perform network maintenance tasks. In most cases, it is a collateral duty. If this continues to be the case, then the OS must automatically perform such tasks as backups, troubleshooting, etc., to relieve the burden of the network manager.

2. Hardware Requirement

Hardware requirements of a typical DII COE system will not only be determined by required functionality, but also by the type of NOS implemented. Traditionally, the

amount of processing needed to perform some of the tasks listed in the DII COE has been performed by Unix workstations, minicomputers, and even mainframes (SNAP, SNAP II). In today's environment, however, this paradigm is changing as processing power gets greater while the container gets smaller and cheaper. The DII COE does list some minimums for both a Unix style workstation as well as a PC. Precise hardware requirements in terms of memory, disk space, etc. is a function of whether the workstation is a database server or client workstation, and whether the workstation is standalone or on a network. [DISA04, p. A-2]

A current implementation of the DII COE, GCCS, is designed to run on the HP 700 Series and Sparc Series workstations. The Navy's TAC-4 contract was awarded to Hewlett-Packard. The minimum configuration for 700 series computer (The HP 9000 Model 712 low-cost Workstation) on the TAC-4 contract comes with:

- 17- or 19-inch pedestal-mount color monitor
- 16 MB - 128 MB of ECC RAM
- 1 GB or 2 GB hard internal disk drives (SE-SCSI)
- One internal 3.5-inch floppy disk drive
- General I/O expansion slot
- Keyboard plus mouse or trackball
- Unix operating system software
- DOS Windowing/Networking Environment

The HP Model 712 also has one integrated LAN interface (IEEE 802.3/Ethernet). Other optional interfaces include a second serial port (RS-232C), an X.25 port, a second monitor port, and a second LAN AUI port.

Additional performance characteristics for the Model 712/80 (the minimal TAC-4 RISC processor platform) include:

- Processor PA7100LC
- Speed 80MHz
- SPECint92 84.3
- SPECfp92 122.3
- Xmark93 8.2
- Graphics Int. Color
- Exp. Slots 1 general, 1 Teleshare [HPAC96]

The requirements for a PC based DII COE system appear to be based more on legacy systems than their workstation equivalents. The DII COE does state that "all [PC] hardware shall be NT-compliant" [DISA04, p. G-10]. The DII COE I&RTS lists the minimum PC workstation requirements as:

- 66 MHz 386 (66 MHz 486 recommended for Windows 95, 90 MHz Pentium recommended for Windows NT)
- 8 MB RAM for Windows 95 (16 MB recommended), 16 MB RAM for Windows NT (32 MB recommended)
- 200 MB disk space required (500 MB recommended)
- 3.5" floppy diskette drive
- LAN Interface card required to access Unix applications
- VGA or SVGA graphics card compatible with Windows NT, and capable of minimum 640x480 graphics in 256 colors
- 15" SVGA Monitor (17" recommended)

Additionally, the DII COE recommends the following equipment to be present in at least enough machines to meet the needs of the individual units:

- 2x speed CD ROM (4x recommended)
- 16-bit Soundblaster® compatible card
- Tape drive for data archival
- HP Laserjet III® compatible laser printer
- Color printer for briefing slides[6]

Over the last decade, most commands have invested heavily in the PC revolution. With this tremendous legacy system in place, it would be a monumental task to switch over to a Unix-based network. Implications of making such a decision must first be considered when selecting a NOS. Some of these problems include, but are not limited to: applications (such as JOPES) not being available to both platforms, user resistance, switching to a different NOS, and commercial alternatives to GCCS software packages.

The minimum requirements set out in the DII COE I&RTS appear to be slightly dated, especially when describing minimum PC requirements. Model 386 machines are rarely acquired anymore, certainly not by large corporations. However, it should be noted that if a computer is needed only for word processing, a 386 machine might be considered sufficient for the near term. This is only valid if a machine only has that one purpose, but as we noted above, in a multi-user environment that the DII COE is expected to perform, single use computers will not suffice. Legacy systems can be left in place if there is no

---

[6] Memory requirements stated here are the minimum for the kernel COE. 32 MB is the minimum for most mission applications; that is for most mission applications not provided by commercial office automation products.

replacement; it is powerful enough to be connected without major alterations or maintenance by the network manager, and it is still functional.

The minimum configuration listed above should not be thought of as an entry system to purchase, only as a stop-gap backwards-compatible interim solution. The minimum configuration should be determined by the unit's workload (server/workstation) and should be within one generation of the latest technology (the Pentium 60 MHz machine is a good example of "old" technology that could be used as a normal configuration). The typical introduction lifecycle of CPUs is 18 months to two years. This means that if you purchase a computer today that is a generation old, it will be two to four generations old after just one tour of duty in the military (three or four years), hard to find support for, hard to purchase software for, and a general time-sink.

C. DII COE OBJECTIVES AND STANDARDIZATION

1. Objectives

In order to meet the operational needs of the DII COE environment, DISA outlines several objectives. The objectives of the DII COE as defined by DISA are:

- **Commonalty**: Develop a common core of software that will form the foundation for joint systems, initially for C4I and logistics systems.
- **Reusability**: Develop a common core of software that is highly reusable to leverage the investment already made in software development across the services and agencies.
- **Standardization**: Reduce program development costs through adherence to industry standards. This includes use of commercially available software components whenever possible.
- **Engineering Base**: Through standardization and an open architecture, establish a large base of trained software/systems engineers.

- **Training**: Reduce operator training costs and improve operator productivity through enforcement of a uniform human-machine interface, commonalty of training documentation, and a consistent "look and feel."

- **Interoperability**: Increase interoperability through common software and consistent system operation.

- **Scalability**: Through use of the segment concept and the COE architectural infrastructure, improve system scalability so that COE-based systems will operate with the minimum hardware resources required.

- **Portability**: Increase portability through the use of open systems concepts and standards. This also promotes vendor independence for both hardware and software.

- **Security**: Improve system security.

- **Testing**: Reduce testing costs because common software can be tested and validated once and then applied to many applications.

These objectives are consistent with those stated in TAFIM. The DII COE states that it will migrate to full compliance with the TAFIM standards profile. The philosophy of the DII COE is that the best way of achieving these objectives is by migrating to open systems.

2. Open Systems

The Gartner Group defines open systems as,

> ...a compliant implementation of an evolving set of vendor-neutral specifications for interfaces, services and protocols, and formats which is designed to enable the configuration, operation, and substitution of the whole system, or parts of the system in a layered systems architecture with applications and/or its components with equally compliant implementations, preferably available from many vendors. [DUNP94, p. 54]

This is just one of many industry definitions of open systems. Most of the definitions of open systems include the idea that open systems are systems that are

portable, interoperable, and adhering to international standards. Portability is the ability to run an application on any systems. It is the ability to take an application from one system and run it on another without having to modify the application. Portability makes it easier to move both work and applications to less expensive or higher capable hardware platforms that could be obtained from a number of different vendors. This ability to port applications reduces the cost for the end user.

Interoperability is the ability to share data across heterogeneous data systems. It also means that the components from different vendors will work together in a system and support whatever application is being run over the entire system. In an interoperable system, different hardware platforms will be connected and be able to communicate. With interoperability, a user will be able to access data anywhere on the network without having to worry where the data is.

An *open* systems architecture is built from a set of vendor-independent, internationally recognized and established standards, as well as a standard application platform. In an ideal world, *open* systems would mean that a user would be able to take components and plug and play them into a system of their choice, just like a consumer would when purchasing a stereo system. In this scenario, a user would be able to mix and match components from multiple vendors according to his desires. Price and performance could be criteria that a user would rely on when considering purchasing components. Vendor independence is an underlying theme of open systems. Independence can be achieved by developing standards that would define the way components from different vendors plug and play with each other.

Normally standards are defined by industry standards, or de facto standards. Examples of the more popular standards organizations that define official industry standards are IEEE and the International Standards Organization (ISO). These bodies often publish standards or specifications on computer interfaces. A de facto standard is a standard that has occurred due to market volume and widespread market acceptance. MS-DOS would be an example of a de facto standard.

Part of the problem in defining a standard is that calling a product standard is common, particularly among vendors, without there really being an industry standard. A product can be measured if it conforms to some standard by its compliance to two measures. The first is whether the product complies with a published interface specification. This can amount to how a software product complies with a benchmark, or an actual compliance test. The second measure is how many instances of the standard exist, or how many instances of the product using the standard in question actually exist.

Another way of measuring "openness" is based on commercial practices. Openness is also associated with how easily a technology is made available to the market, and at what cost. If a company freely documents its software, source code, or even its hardware design, it is a more open company than one that has expensive or a restrictive licensing practice. An open standard is one that is based on cost rather than value. For many years, and perhaps still today, Unix was considered by many to be an open standard. Unix was considered open because it could be licensed by anyone who purchased it during its developing days. Unix source code was also readily available. This led to the development of many Unix environments. One sign of an open standard is one in which you can point to more than one implementation of it. Unix is one example of an operating

system that has this characteristic. This is contrary to proprietary operating systems such as Windows, MacOS, and OS/2. The future developments of these operating systems are controlled by a single company. When specifications are openly available to everyone, they are open. When they are protected and private, they are proprietary. [DUNP94, p. 22-23]

Standards then are at the basis of the concept of open systems. They are formed to help eliminate confusion and achieve consensus among the many interests in the computer industry. In the government, as well as private industry, standards are important to those responsible for designing information systems of the future. Standards are intended to promote the following:

- Lower-cost, higher quality products and services.
- Open interfaces that increase the potential for interoperability.
- Multiple sources for components - a form of insurance.
- A common set of benchmarks for evaluating alternatives.
- A greater selection of common solution elements from multiple vendors.
- The potential to exploit interoperability at various levels within the system architecture. [DUNP94, p. 25]

Achieving open systems will require organizations (including the government) to continue, or start adhering to some set of agreed upon standards. By doing so, there will be an environment that is multivendor, competitive, and multi-sourced.

Standards organizations, like the IEEE, are currently playing the role of publishing uniform operating standards for open systems. The IEEE and other organizations have concentrated on program interfaces to support some degree of application portability.

Typically, a lowest common denominator is agreed upon as a standard interface. This lowest common denominator provides the minimum amount of functionality, and vendors in turn embellish this standard to meet the needs of the market place. While the standards that are published are often the minimum standards, they provide a stability for organizations trying to achieve *open* systems, as well as a litmus test for measuring the compliance of vendors with *open* systems. Standardization organizations will play a vital role in defining open systems in the future. [KAMA94, p. 12]

3. Open Systems in the DOD

The role of *open* systems in the development of information systems dates back to 1992. In a memorandum dated February 12, 1992, the former Director of Defense Information, Paul Strassmann stated, "Implementation of open systems is essential to reducing systems costs, improving information sharing and interoperability, streamlining acquisition times, and enabling the other improvements envisioned through the application of corporate information management." [STRA92] Strassmann's directive states that all new systems development and major modernization plans are required to use the DOD's TRM as the guideline to select appropriate standards for implementation and future systems planning.

According to the TRM, a common open systems environment will provide a basis for the development of common applications and facilitate software reuse. *Open* systems will promote portable applications, which will allow activities to be able to upgrade their hardware base with technology with minimal impact on operations. Interoperability will be improved by implementing a common infrastructure through standardization. Vendor independence will occur by acquiring only interchangeable components and supporting

non-proprietary specifications. The TRM states that by applying all of these principles, there will be a reduction in life cycle costs. It also states that this cost will be reduced by the eventual replacement of stovepipe systems with interconnected *open* systems. These interconnected systems will be able to share information and will reduce the redundancy and data duplication in current systems. [DISA03, Vol. 2 p. 2-1]

The goal of the standards profile presented in the TRM is to have DOD information systems achieve an open systems environment. The DOD selected the set of standards based on several criteria, including level of consensus, product availability, completeness, maturity, stability, de facto usage, and product limitations. [DISA03, Vol. 2 p. 3-1] As mentioned, the intent of the DII COE is to be fully compliant with the standards profile of TAFIM's TRM. GCCS, fully TAFIM compliant, subscribes to many of the applicable standards in the TRM. Appendix A, the GCCS COE as-built standards profile, shows several of the standards adopted by GCCS. The list shows the dependence on standards of different service areas of GCCS. It demonstrates the DOD's reliance on standards in the development of DII COE based systems.

4. DII COE/TAFIM defined standards

As discussed above, in order for an *open* system to be achieved, a set of standards must be agreed upon. POSIX is one such standard that the DOD has adopted and is at the heart of the DII COE. Some other prominent standards include communications standards (TCP/IP, FTP) and GUI standards (MOTIF).

a. POSIX requirements

In defining the standards of the DOD common architecture, the TRM adopts the framework of the IEEE POSIX P 1003.0 Working Group. POSIX stands for Portable Operating System Interface for Computing Environments. The last letter X denotes POSIX Unix origins. POSIX refers collectively to a number of standards specifications. It is an interface standard for portable operating systems being developed by a number of committees organized by the IEEE. POSIX 1003.1 details only basic operating services.

The goal of these committees is to improve industry-wide portability and interoperability. There are two types of standard interfaces specified in POSIX: the APIs, and the external environment interfaces (EEIs). APIs are the procedure calls made to the application platform. The application platform is the computer in which the application platform is running, as well as the OS. APIs provide for source code portability. The EEI refers to external entities with which the application platform exchanges information. This could include the end-user, and physical devices such as terminals, printers, and networks. EEIs generally are in the form of communication protocols and provide for interoperability. POSIX is like a list of standard commands that an OS should be able to perform in a given manner when called upon to do so by an application program. If software applications use these calls (and no non-standard ones), then it should behave the same on any NOS that supports these calls (i.e., is POSIX compliant, assuming the same version of POSIX compliance is adhered to). This would eliminate the need for software developers to produce several different versions of the same application to accommodate different operating systems.

Portability and interoperability standards like those listed in POSIX, however, are not sufficient for a complete *open* systems environment. POSIX offers a building block that was chosen by the DOD, and subsequently complimented with several other computing standards.

b. Other standards

POSIX alone does not guarantee complete open system concepts like interoperability and portability. TAFIM relies on a number of other standards to help define other information system service areas. These standards include:

- ADA 95
- Motif
- TCP/IP
- X.400/X.500
- 802.2/802.3/802.4/802.5

These standards help further define features that are not covered by POSIX. The X-Windows system protocol is a good example of how this is done. X-Windows "specifies how graphic primitives can be communicated between an application program and graphics software." The interoperability between POSIX compliant platforms does NOT guarantee that source code will run on two different machines running POSIX compliant NOSs, because the two system might use different library functions to produce the X-Window protocols. [KUHN91, p. 37]

It is important to note that the DII COE was developed by DISA, and thus requirements such as those for POSIX were included because of DISA's belief that this

was the best method to achieve *open* systems in the DOD. We have adopted DISA's guide for the DII COE in our evaluation of how the NOS fills the needs of the infrastructure that DOD will put in place in the future.

III. TECHNICAL ARCHITECTURE FRAMEWORK FOR INFORMATION MANAGEMENT (TAFIM) OBJECTIVES

A. IMPROVE USER PRODUCTIVITY

1. TAFIM's definition of the objective

TAFIM's definition of improving user productivity will be realized by applying the following principles:

- **Consistent User Interface.** A consistent user interface will ensure that all user accessible functions and services will appear and behave in a similar, predictable fashion regardless of application or site. This has the benefit of simplifying training, facilitating the development of future applications, improving ease of use across applications, and promoting application portability.

- **Integrated Applications.** Applications available to the user will behave in a logically consistent manner across user environments. Support applications, such as office automation and electronic mail, will be used as an integrated set with mission area specific applications.

- **Data Sharing.** Databases will be shared across DOD in the context of security and operational considerations. Concepts and tools that promote data sharing include adherence to standard database development rules, the use of DOD data dictionary and software reuse libraries, and strong DOD commitment to resource sharing. [DISA03, Vol. 2 p. 2-1]

2. Interpretation of objective

a. Consistent user interface

We interpret TAFIM's objective to improve user productivity through the use of a consistent user interface as providing a graphic user interface (GUI) that users are comfortable and familiar with. We say graphic because that is what the majority of DOD

personnel are using today, and the technology for providing GUIs is relatively mature. However, improved technology has introduced voice-activated software, which has the potential to become an industry trend in the future, and may have several productivity benefits over GUIs. A consistent user interface will provide users with the familiarity and "look and feel" among the NOS and its applications. This consistent "look and feel" facilitates transitioning to new or upgraded versions of the NOS with a relatively flat learning curve, thereby reducing training costs associated with that transition. A consistent user interface provides a common set of pull-down menus, common toolbars, and buttons. To the extent possible, the user interface operates in the same consistent manner regardless of application. Familiarity of system utilities, such as a file or print manager and fast easy access to online help, is also essential.

b. Integrated applications

TAFIM's objective to improve user productivity by integrating applications provides users with a set of software packages capable of meeting all their business, professional, and tactical support needs. Again, consistency among the applications in user interface, communications protocols, and integrated use of applications is an essential element in improving user productivity. The NOS provides the common foundation and vehicle for integrating applications. For example, being able to use database and spreadsheet data together in a word processing document, or being able to share document data in a presentation graphics application, typifies the concept of integrating applications. Software integration (e.g., a suite of integrated applications) provides the user with a familiar and consistent operating environment across a wide spectrum of

programs (database, spreadsheet, word processing, management information systems, etc.) that will improve user productivity.

The NOS can and should provide mechanisms for applications to communicate (e.g., share data) between each other. These mechanisms will permit application integration among a variety of vendors, and not necessarily be limited to a single vendor's suite of applications.

c. Data sharing

Establishing standardization for data sharing improves user productivity by creating, maintaining, and updating both centralized and distributed databases, which DOD users can access and utilize. For administrative uses, military career and personnel data can be maintained at a centralized database. Distributed access of information should be accessible by authorized personnel maintaining and using those records. Service member career information (e.g., pay records, promotion information, medical and dental records, etc.) can be accessed, downloaded, and updated, and remains available even when personnel change jobs or location.

For tactical support and intelligence support applications, database architecture becomes more complicated, but must conform conceptually to both centralized and distributed database models. Data for tactical systems must be centralized at the local command, where sensors can immediately update information, and then local users can immediately retrieve it. For shore-based analyst personnel, data can be distributed throughout several national level data centers where real-time access to data is not as essential. Through consistent and standardized data sharing techniques provided by

the DBMS, data centers can receive data updates from theater units on a periodic basis, as well as share national level information to theater units as requested.

At the DBMS level, data sharing is achieved through the use of standardized data formats. While standardizing databases on a common DOD-wide data dictionary is essential to sharing data through a database management system (DBMS), the NOS must also implement data handling controls (packet size, parity checking, etc.) to share data across the WAN and across multiple applications in a standard predefined way. This way the DBMS, with the support of the NOS, can consistently and efficiently pass data between applications and systems, both local and remote.

Establishing common data sharing mechanisms at the DBMS level, as well as the NOS level, are key elements in improving user productivity. Data sharing will save time in database entry, improve maintenance of numerous databases, increase information exchange across applications, and improve the quality and content of the information. At the NOS level, implementing a consistent data sharing mechanism allows data to be efficiently exchanged between locations.

The data sharing element of improving user productivity is closely tied to developing integrated applications and improving interoperability, which will be discussed in later sections of this chapter. It is expected that most users would have been exposed to the Microsoft Office application suite or the Corel Wordperfect Suite software package, which both share data seamlessly between word processors, databases, presentations graphics, and spreadsheets. In tactical support applications, equivalent data sharing mechanisms allow applications to utilize data from one application, say a standardized mapping program (Chart Service), with the JMCIS track file. This provides

a geographic display of a tactical area overlaid with the current positions of vessels at sea. Incorporating the data sharing mechanism into the NOS provides this service at the foundation of the information system architecture and ensures compatibility across the information system.

3. Analysis of the Unix architecture

The initial work on the Unix operating environment began in 1969 to provide an OS that would support programming research. The timing and purpose for the original development of Unix had a strong influence on the design of the Unix NOS architecture. The original variant of Unix was owned by AT&T, but due to antitrust restrictions, AT&T was forced to place Unix onto the "freeware" market. AT&T Unix was used primarily at universities on minicomputers and mainframes. Over the years, Unix has grown in size as well as in its repertory of tools and utilities, and has spawned numerous variants of the original Unix. Throughout the years, university research and research projects have developed new variants, and modifications of the original Unix. In many respects, this evolution of Unix in the education and research environments forged many improvements over early variants resulting in increased system performance, optimization of the OS kernel, interfaces, etc. This has resulted in a Unix OS which today is a widely used time sharing system for use in both commercial and DOD applications, and is currently the basis for the DII COE. This evolution has, however, resulted in significant incompatibilities between different Unix variants. The Unix variants, HP and Solaris, are the only approved variants of the Unix NOS for the DII COE. Figure 7 depicts the evolution of the Unix operating environment.

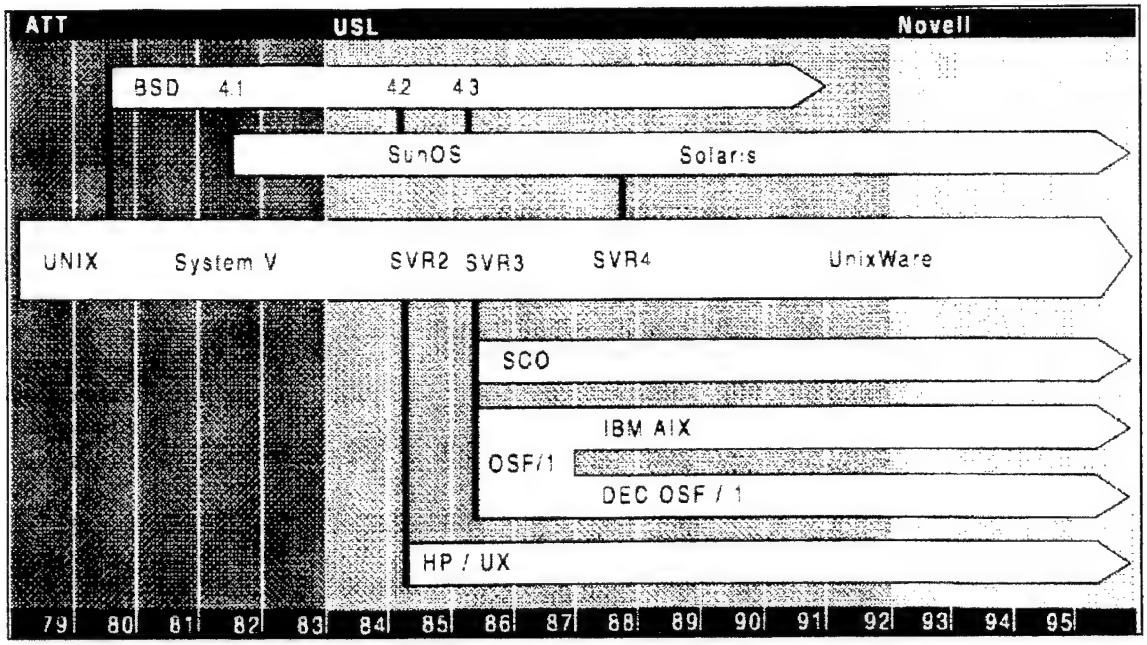

Figure 7. Unix Evolution [MICR03]

a. The user interface

While the objectives in this chapter examine the architecture of HP-UX v9.01 as the choice of NOS for the DII COE, it is important to recognize that Unix is not "Unix". Keeping this in mind, the native user interface to HP-UX v9.01 is the command line interface similar to Microsoft's Disk Operating System (DOS). While this user interface does not achieve the intent of the TAFIM objective for improving user productivity through the use of a GUI, most variants of Unix, specifically HP-UX v9.01, support the use of a GUI windowing system based on the X-Window system.

The X-Window system, like Unix, comes in many "flavors". The DII COE specifies Motif, from the Open Software Foundation (OSF) as the standard windowing system. Motif is a standard for providing the DII COE running the Unix NOS with a GUI, and meets the requirements of TAFIM. While the X-Window system and Motif standard provide the specifications for implementation of the GUI, DOD must adopt one

or more proprietary Motif compliant X-Windowing GUIs, much like DOD must adopt a proprietary NOS which is POSIX compliant.

The X-Window/Motif GUI is installed and operates much the same way that Microsoft Windows v3.1 operates on top of DOS in most PCs. Similarly, applications designed to run in a GUI environment must run on a compatible Unix NOS running the X-Windows GUI user interface. Compatibility between Unix NOSs is limited to those which comply with the POSIX/IEEE 1003.1 open systems standard (discussed later). X-Windows includes a separate GUI-oriented Application Program Interface (API), which is defined by the Motif standard and acts as an abstract interface to the services and protocols offered by the Unix NOS through a set of function calls. Applications use the function calls available in the X-Window interface to gain access to the OS's services in a graphic environment instead of the Unix command line interface [NORT91]. It should be noted, however, that while Motif and the X-Window GUI are POSIX compliant, application programs running on the Motif GUI must be programmed using the POSIX API and the Motif GUI API in order to function on a specific NOS and GUI. This is a requirement because the POSIX API defines how applications interact with the NOS, and the GUI API specifies how the NOS interacts with the display system.

Motif is a widely-accepted set of user interface guidelines developed by the OSF around 1989, which specifies how an X-Window system application should "look and feel." OSF/Motif includes the Motif Style Guide specifications, which details how a Motif user interface should look and behave to be "Motif compliant."

The Motif's style guide allows each workstation using the Unix NOS/X-Window GUI to be configured to a wide variety of operating conditions, including low

light, blue light, red light, normal light, and outdoor environments. This is important because it supports the wide spectrum of end user needs discussed in Chapter II. Motif conformance allows users to customize their desktop and workspace for individual preferences.

b. Application integration

In support of TAFIM's goal of integrating applications, the Unix operating environment provides the platform upon which development of JMCIS, GCCS, and other service-equivalent information systems can be incorporated into the DII. Integration of tactical support applications is achieved by assigning responsibility to services and software development organizations to particular software segments of the total information system.

The key to this multiple department development program is adherence to a common set of specifications, which permits interoperability. The Unix operating environment, being POSIX compliant, provides the POSIX API, which gives developers the ability to code their respective segments to integrate into the overall information system. This integration is demonstrated by the GCCS running the JMCIS segment. In execution, JMCIS calls on data from the JMCIS maritime track database, and overlays track positions and data over the Army's Chart Service, which provides the "map" of the desired area. While program development must include code that allows the integration of applications, it must be provided with a vehicle. This vehicle is the Unix NOS, which acts as the foundation (a consistent base operating environment) of the information system to integrate applications in a common environment.

c. *Data sharing*

Unix provides the mechanisms for application integration and data sharing through its core architecture. Figure 8 shows the Unix core architecture. Data sharing between applications and processes is invoked in the Unix operating environment through interprocess communication (IPC) routines. The Unix system provides the following mechanisms for performing IPC:

- Unnamed and named pipes.
- Shared memory
- Message queues.
- Semaphores
- Signals
- Sleep and wakeup calls

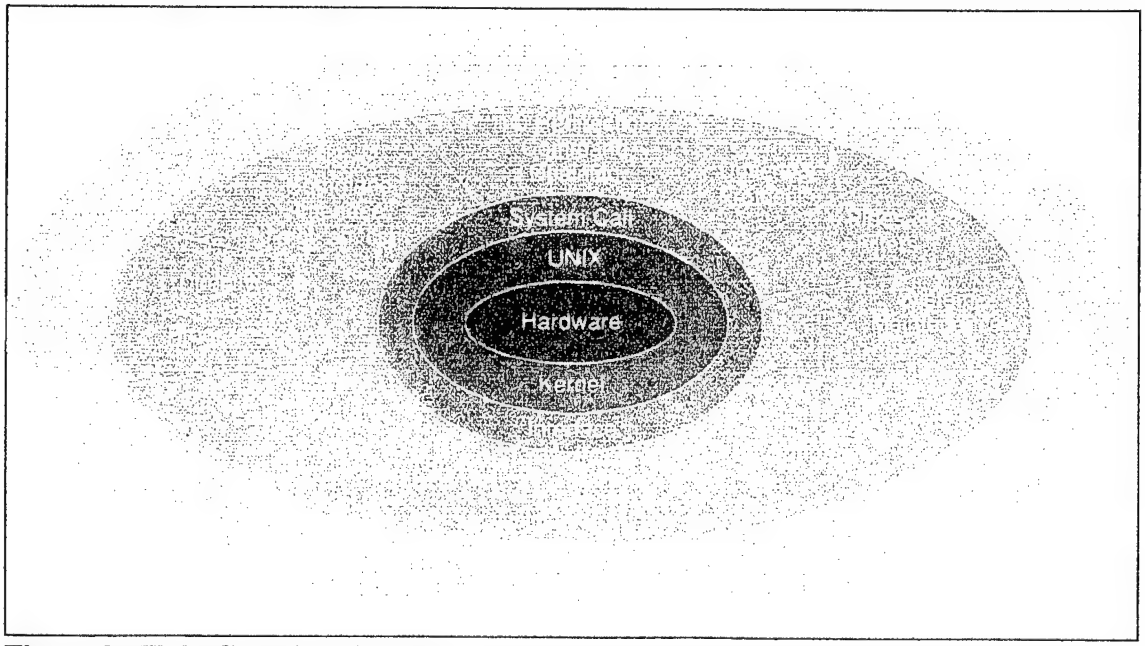

Figure 8. Unix Core Architecture

While the architecture for each of these mechanisms resides in different modules of the NOS, they all reside within the Unix NOS, and operate together to provide data sharing between applications. A detailed description of how each of these mechanisms operate is not essential in understanding that Unix provides data sharing services. However, the reference section contains sources which describe each mechanism's operation in detail. The key concept here is that the Unix NOS provides several IPC mechanisms that allow programmers to integrate applications running on the Unix NOS, as well as providing the vehicle for data to be shared between applications both locally and remotely [ANDL90]. While the Unix implementation of data sharing is adequate, it does not offer the rich interaction of more recent data sharing technologies.

4. Analysis of the Windows NT architecture

Windows NT is a NOS which was first introduced into the commercial market in 1991. Since then, Windows NT has undergone two major upgrades, which has kept pace with the rapidly changing information technology industry. With the release of Windows NT version 3.51, and the recent release of version 4.0, Microsoft Windows NT has arguably become the most interoperable NOS in the market today.

Windows NT was not the first NOS designed to exist on both local area and wide area networks, but it was, unlike Unix and other OSs, built from the ground up with connectivity in mind. The Windows NT design began with two sets of requirements: market and design. Under the market requirements, Windows NT provides:

- Portability across families of processors, such as the Intel 80X86 and Pentium lines
- Portability across different processor architectures, such as CISC and RISC

- Transparent support for single-processor and multiprocessor computers
- Support for distributed computing
- Standards compliance, such as POSIX
- Certifiable security, such as C2 and F-C2, E3

Additionally, the Microsoft development team established the following design goals for Windows NT:

- Extensibility
- Portability
- Reliability
- Robustness
- Performance
- Compatibility

All of these requirements and goals help make Windows NT interoperable with a wide range of legacy systems in the market place, government, and the home, as well as provide consistency and commonality. [FEIB95] Figure 9 describes the Windows NT architecture.

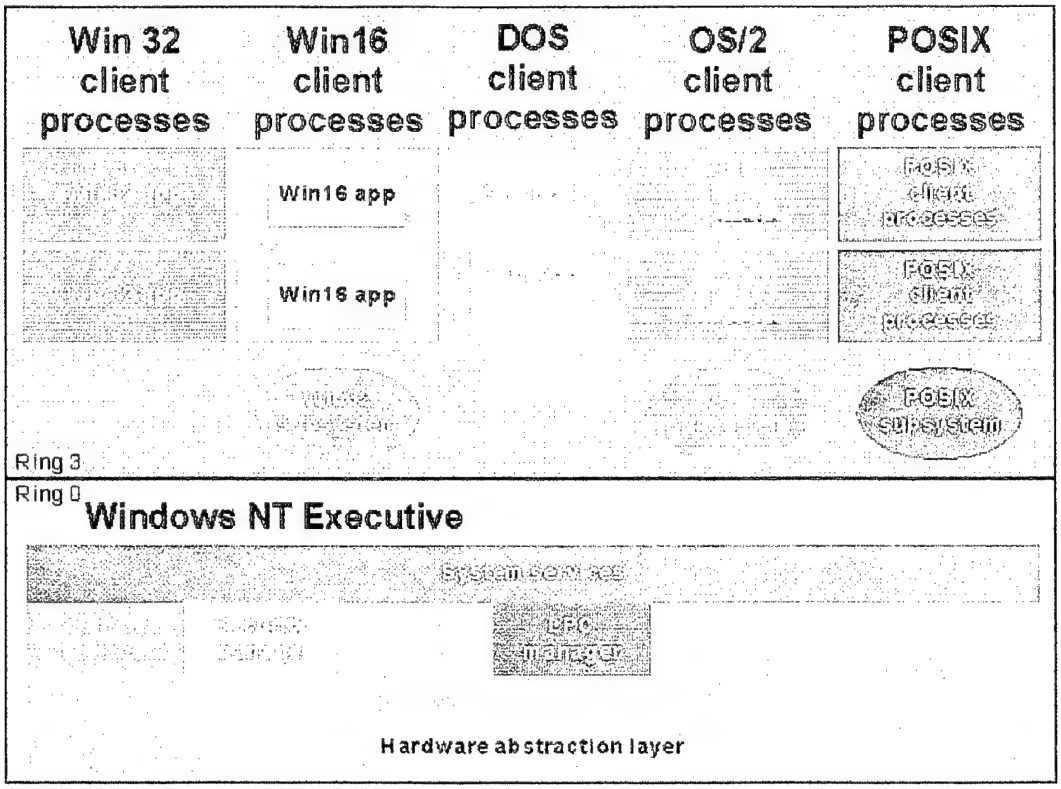

Figure 9. Windows NT Architecture

a. The user interface

The Windows family of OSs has incorporated a graphical user interfaces (GUI) into all of its products. The Windows GUI has changed over the years, culminating in the new user interface first introduced in Windows 95, and incorporated into the recently released Windows NT version 4.0. Design changes to the Windows NT GUI are the result of Microsoft's visual design group, tasked with making the Windows GUI look and behave in a consistent and similar manner across applications [KING94]. The visual design group at Microsoft is chartered to make the Windows NT GUI more document-centric in order to enhance end user productivity.[7] A document-centric approach means

[7] The document-centric appearance was first introduced by Apple Computer Corporation in their line of Macintosh operating systems.

that users concern themselves only with documents and not with program files. This makes the NOS responsible for maintaining the relationship between data of a particular format and the application that can manipulate the data.

A user who is unfamiliar with the details of a particular operation or application may first seek visual guidance (looking for cues such as dialog boxes, shaded pull-down menus, and help menus), while navigating through a sequence of actions aimed at providing the desired result. The Windows GUI tends to reduce the learning task associated with a new application as compared with character oriented OSs like DOS, by presenting access to many standard operations in the same manner. While this concept is not much different than the X-Window/Motif GUI found on Unix systems, the Windows NT visual design team works specifically to optimize the following GUI characteristics:

- **Consistency.** Does the user always do the same operation in the same way? Does the user gain access to similar operations using the same keyboard or mouse inputs, guided by similar visual cues?

- **Usability.** Does the interface allow the user to do simple operations simply and complex operations within a reasonable number of operations? Forcing the user to go through awkward or obscure input sequences leads to frustration and ineffective use of the system.

- **Easily Learned.** Is every operation simple enough to be remembered easily? What the user learns by mastering one operation should be transferable to other operations.

- **Intuitive.** Is the interface so obvious that little or no training or documentation is necessary for the user to make full use of it? This aspect of a GUI is the holy grail for interface designers.

- **Extensible.** As hardware gets better and faster – for example, as common screen displays achieve higher resolution or new pointing devices appear – can the interface grow to accommodate them? Similarly, as new application categories become popular, does the user interface remain valid?

- Attractive. Does the screen look good? An ugly or overpopulated screen will deter the user and reduce the overall effectiveness of the interface [KING94].

As with the X-Window/Motif system used by Unix, the Windows NT GUI can be customized and configured to conform to the operational conditions of the current environments defined in chapter II (low light, red light, etc.). Additionally, given Windows NT's POSIX compliance (discussed later), it is possible to run the X-Window/Motif GUI from within the Windows NT NOS to access and run Unix applications [PARA96]. Finally, the Windows NT GUI API is incorporated into the native Win32 API specification, thus integrating the GUI and OS APIs into a single specification, instead of the two API specifications required by the Unix POSIX and Motif GUI standards.

b. Application integration

Windows NT supports integrated applications. Numerous integrated applications exist in the market today that are designed for the Windows NT NOS. As stated above, the preponderance of PCs and Windows family of OSs offers a significantly greater number of integrated applications specifically targeted for home users and corporate users. Microsoft Office suite and Corel Wordperfect suite demonstrate the maturity and ease of integration across applications in their respective suites. Windows NT provides the same consistent base operating environment as the Unix NOS with which applications can be integrated.

c. Data sharing

Microsoft's strategy for integrating applications and improving cross-application data sharing is built around a technology called Object Linking and Embedding (OLE). OLE provides the data sharing and IPC to improve cross application functionality and user productivity. With OLE compliant applications, users get the following capabilities:

- **OLE Documents:** OLE documents improve the process of creating documents and the content of business documents. OLE documents can contain any type of information, including text, spreadsheet tables, pictures, graphics, video, sound, or any information object. The information contained in an OLE document can be created using any supporting OLE-enabled application, such as a spreadsheet application, graphics application, or multimedia application. These applications can be supplied by different software vendors who support OLE, because OLE components work together. OLE documents not only enhance user productivity, they also enable users to communicate their ideas more effectively. As a user edits an OLE document that contains different types of information, the specific tools necessary to edit the different types of information are automatically made available to the user within the context of the document. This is called Visual Editing. With OLE Linking, a document can contain information that maintains a data link to another document.

- **OLE Drag and Drop:** OLE drag and drop allows users to directly exchange information between applications, without having to save files to disk or converting information to different formats. For instance, a user can point to an embedded spreadsheet in a document and drag it over to another document in another application. By making data exchange graphical and intuitive, users can increase productivity.

- **OLE Controls:** OLE controls are OLE-enabled software components that users can purchase to extend and enhance an application's functionality. Users can utilize OLE controls in custom or off-the-shelf OLE-enabled applications. Most popular development environments, including the next version of the Microsoft Visual Basic programming system, will support OLE controls as an efficient means to build business applications using high-quality, prefabricated software components.

- OLE Automation: Automation enables applications to provide command sets that operate within and across applications. For example, a user can invoke a

command from a word-processing program that sorts a range of cells in a spreadsheet created by a different application. [PARA96]

Incorporating OLE technology into the OS and applications is not without disadvantages. OLE support to an application is an extremely complex engineering project. New development tools and methods will ultimately reduce the complexity and cost of OLE implementation [KING94], but the near-term outlook for incorporating OLE technology is that it is likely to complicate and prolong application development, as well as reduce initial development efficiency. This will be discussed in detail in the next objective.

Windows NT also supports mechanisms in its core architecture similar to those found in the Unix core architecture. This support includes shared memory, pipes, semaphores, and message queues [CUST93]. While their implementation in the NOS architecture is different than its Unix counterpart, they serve to provide the necessary IPC between applications to permit data sharing, even when OLE technology is not implemented in the application.

5. Summary of findings

Both the Unix and Windows NT NOSs provide several mechanisms which work to improve user productivity. Although neither NOS provides significantly better mechanisms to achieve the user productivity tenets that TAFIM outlines, we believe that Windows NT employs a more popular user interface and a more robust method for IPC. This makes it a better choice of NOS for improving user productivity. Specific summaries for each of the tenets of the objective are provided.

a. The user interface

From the perspective that Windows NT achieves the user interface characteristics described above, it enhances user productivity as well as, and in some respects better than, the X-Window/Motif system of Unix. While Unix and Windows NT both incorporate a GUI, Windows NT has the added benefit of its GUI being built-in to the NOS. The Unix GUI runs as a third party shell on top of the Unix character-based NOS. Since improving user productivity is partially based on how familiar the user interface is to its users, then Windows NT offers an advantage which X-Window/Motif does not. It is estimated that there are more than 2.5 million Unix systems in use today. The Microsoft family of OSs, complete with the Windows GUI, however, has an installed base of more than 60 million on the PC. [PARA96] Many personnel entering military service today have experience with the Windows OS. Some military personnel now use Windows OSs on desktop PCs for administrative uses, as the Services become increasingly reliant on personal computers. Microsoft continues to adjust its GUI through upgrades to the Windows family of OSs. The Windows GUI remains easy to learn and use because new versions generally stay along the same theme of the Windows Environment.[8]

8 Windows 95 represented a major change to the Windows family GUI. Windows NT 4.0 incorporates the Windows 95 GUI, establishing consistency in the GUI along the new user interface. For the foreseeable future, Microsoft NOS's will most likely incorporate the newer Windows 95 GUI or slight variations of it.

b. Application integration

Both NOSs provide the operating environment and vehicle to provide broad based support for integrated applications. Because the Windows NOS family enjoys an installed base of more than 24 times the estimated number of Unix systems, the commercial development of integrated applications will follow the market trends and consumer demands. This development provides DOD with a wide range of options for various integrated applications to support the needs of the end user using both the Unix NOS and the Windows NT NOS. For custom application development, both NOSs provide a common foundation for building integrated applications; however, the larger Windows family installed application base gives Windows NT a slight advantage over Unix.

c. Data sharing

Both Unix and the Windows NT NOSs provide mechanisms to support data sharing. Windows NT provides both its own proprietary technology, OLE, and supports the more fundamental IPC mechanisms to provide cross vendor and cross application data sharing. The Windows NT technology offers a richer and more intuitive set of mechanisms which will ultimately translate to improved user productivity, provided that application developers incorporate the OLE technology into their applications. While OLE improves user productivity, it demands more complexity in the application program, and requires users to learn and understand how to use that technology to accomplish tasks using OLE.

B. IMPROVE DEVELOPMENT EFFICIENCY

1. TAFIM's definition of the objective

TAFIM's definition of improving development efficiency will be realized by applying the following principles:

- **Common Open Systems Environment.** A standards-based common operating environment, which accommodates the injection of new standards, technologies, and applications on a DOD-wide basis, will be established. This standards-based environment will provide the basis for development of common applications and facilitate software reuse.

- **Common Development.** Applications that are common to multiple mission areas will be centrally developed and acquired.

- **Use of Products.** To the extent possible, hardware-independent, non-developmental items (NDI) should be used to satisfy the requirements in order to reduce development and maintenance costs.

- **Software reuse.** For those applications that must be custom developed, incorporating software reuse into the development methodology will reduce the amount of software developed and add to the inventory of software suitable for reuse by other systems.

- **Resource Sharing.** Data processing resources (hardware, software, and data) will be shared by all users requiring the services of those resources. Resource sharing will be accomplished in the context of security and operational considerations. [DISA03, Vol. 2 p. 2-2]

2. Interpretation of objective

Common open systems

A common *open* systems environment is what DOD (and the government in general) is trying to achieve with the DII COE framework. A common operating environment is suppose to achieve maximum competition in the market place by providing standards that are approved by standards committees, such as the National Institute of Standards and Technology (NIST) or the IEEE. It is important to recognize that these

standards are open to independent implementation. Any vendor complying with given standards can develop and market software applications that will run on any system that complies with the standard. The current focus in the computer market place is on standardization.

But standards are not as "standard" as one might think. While standardization is a common trend in DOD and the private sector, there are many "standards" with which a system must comply to enable an information system to operate properly in a given environment. The example of how the DII COE uses the POSIX 1003.1 standard for NOS to application interaction, and the Motif standard to define the NOS to GUI interaction, demonstrates that POSIX compliance in and of itself is not sufficient to provide the end user with a working application or information system. Table 1 provides a sample of some of the more common *open* systems standards which exist today.

• POSIX .1	• CDE
• POSIX .1b	• X11
• POSIX .1g	• Motif
• POSIX .2	• HP's VUE
• POSIX .1e & .2c	• Sun's OpenLook
• POSIX .5	• OpenDoc
• XPG4 UNIX	• XPG4 Base
• XPG4 V2	• GCCS COE

Table 1. Open Systems Standards [ROYS96 p. 8]

As Table 1 and Figure 10 show, there is no single *open* systems standard, nor is there one encompassing standard covering all implementation procedures or APIs. Standards, whichever ones have been established for a given information system, must be

implemented together to provide an environment with which application development can be accomplished. Given a specified set of standards, application developers must then program with every standards' API set in mind.

While the use and development of *open* systems standards is beneficial in improving development efficiency, care must be taken to select standards at the appropriate level and specificity to be used in both the information system and the COE. As any standard becomes better defined and more specific, the application developer is provided with a more regimented set of implementation procedures. In effect, more descriptive standards give less for the developer to interpret from the standard, and provides more consistency in application development. A more descriptive standard limits the developers ability to apply "non-standard" (creative) implementation of the API or application. While it appears that the developer is more constrained in how they implement or use APIs, *open* systems standards ensures implementation consistency across applications and information systems. Figure 10 demonstrates the different levels of *open* systems standards.

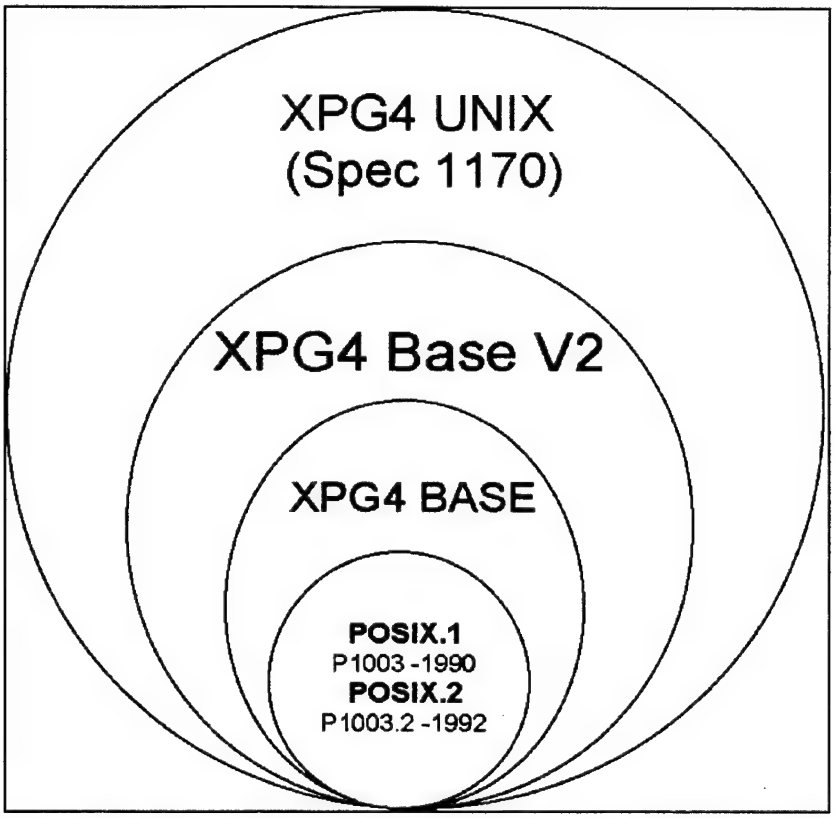

Figure 10. Scope of Open Systems Standards [ROYS96, p. 9]

The NOS provides a foundation for standards implementation. It is interesting to note that while POSIX is an *open* systems standard, it is based on existing Unix APIs. Ideally, an *open* systems standard which is truly vendor independent should be independently developed, and not based on any specific NOS or NOS variant. This ensures vendor independence.

While open systems standards are required by TAFIM, DII COE, and the mandate of DOD for information systems development, there is another established type of standard which achieves TAFIM's objective to improve development efficiency. That standard is the selection of a particular proprietary NOS. Adoption of this method as a standard supports development efficiency, but does not promote vendor independence or competition, as discussed later. It is therefore seldom considered a means of improving

development efficiency. Given this, the argument for establishing an *open* systems based standard to improve development efficiency is achieved not because of the particular standard that is adopted for systems development, but because the fact that a standard has been adopted and used to facilitate application development.

b. Common development

By applying common development strategies to systems development, TAFIM expects to centrally develop and acquire applications from single points of contact. The benefits of common development strategies are similar to the benefits of moving from flat file database structures to relational structures. The database analogy reduces the redundancy of data and eliminates the need to key identical (or similar) data into separate data structures and databases. Common development strategies eliminate the need for organizations in the DOD to develop routines or applications which functionally serve the same purpose.

The GCCS has made significant strides in improving overall system development efficiency by using common development strategies. DISA, the primary development agency of GCCS, has identified and assigned common software modules and applications to provide functionality across service boundaries. The Army provides the mapping data and service routines for all geographic displays. Navy, Marine Corps, Air Force, Coast Guard, and intelligence agencies use these routines with their respective applications and do not need to develop service specific mapping routines. If a software application shares a common use with another, then their development and acquisition should also be together. This reduces the number of software applications and routines

that need to be developed for DOD, and allows software to be developed more quickly and efficiently. Common development strategies improve the overall quality of software used by DOD, and development is completed in a more timely and efficient manner. Defining a standardized NOS environment allows applications to be developed to a common architecture and standard.

Common development techniques require that specific open systems standards be defined. This provides the base set of APIs for application development and provides the basis for software compatibility. Without adherence to the same open systems standards, it is unlikely that segments or applications relying on modules from other applications would function properly. The need for adherence to open systems standards to ensure functionality across applications is best demonstrated by looking at the many variants of Unix. While many programs have been developed for Unix OSs, several applications using an API set for one variant of Unix may not function properly on a different Unix variant. Successful integration and sharing of applications (or application segments) using common development techniques demands that their development use the established baseline standards and APIs.

c. Use of non-developmental products

The use of non-developmental products effectively uses commercial off-the-shelf (COTS) and government off-the-shelf (GOTS) software in DOD information systems. COTS software is software that DOD purchases from commercial vendors to fulfill specific DOD needs, and is generally not modified for DOD use. GOTS software is software that the DOD or other government agencies have developed or retain the

licensing rights to. GOTS software exists as both whole applications, and as software modules which are available to government organizations for reuse. COTS and GOTS software is, in most cases, independent of special hardware or customized development. This enables the software to comply with established standards for the particular information system or computer architecture. The question for the DOD program managers is whether to use GOTS or COTS software as the primary option.

It is reasonable to assume that GOTS is more readily available and less costly option, since it is "owned" by government entities. This assumes that the degree of adaptation or re-engineering of GOTS modules is small or non-existent. Reusing DOD GOTS software (assuming it was developed with the DII COE framework in mind) would be consistent with the common "look and feel" of DII COE applications by virtue of DOD wide application development consistency efforts. GOTS software can be more costly than COTS when it requires substantial modifications to provide the necessary functionality for the information system.

COTS should follow in preference, filling in the gaps that are not adequately fulfilled utilizing GOTS software. COTS software provides for a wide range of needs, particularly in automation and administrative applications. Here commercial development efforts have targeted sophisticated software development towards filling the demanding needs of both corporate America and the home user.

The final option for DOD program managers is to develop the software either "in house" or to outsource to a commercial vendor for special purpose development. This last option is usually more costly and should be avoided when possible to reduce or eliminate redundant development efforts and save limited financial resources.

Outsourcing and "in house" development offers the advantage of being able to customize application development to a specific need, which may not exist in COTS or GOTS software.

The issue of using non-developmental products as a method of improving development efficiency requires managers to weigh the perceived benefits of readily available COTS software and easily modifiable GOTS software against the cost and development time of outsourcing, or "in house" development. It may be better to have an immediate or short term GOTS or COTS software application with 80 percent functionality than an expensive new application development effort which provides an application with 95 or more percent functionality and a relatively long developmental time.

The problem of establishing rules specifying using GOTS before using COTS does not allow the flexibility needed by DOD in choosing the best mix of software. There is a price-productivity tradeoff between the acquisition cost of COTS and the development cost of GOTS. For example, Applix (a DOD Unix-based application similar to Microsoft Office currently on GCCS) is not as user friendly or full-featured as most commercial office suites. As a result, the extra dollars spent on a familiar COTS office suite may be well worth the improved productivity likely to occur by using a familiar and fuller featured software program.

Using COTS products relies on the NOS to provide the foundation with which applications can operate in. The larger the pool of software products to choose from and the greater the competition between vendors, the better the chance end users have to get the desired program and functionality. This implies that the DII COE

framework, and the NOS, should be able to run applications compatible with the largest installed commercial base NOS.

While POSIX compliant programming does not directly improve development efficiency, it does allow developers to provide a single product capable of running on a wide variety of systems (those conforming to the POSIX standard). This makes POSIX compliant applications both hardware and NOS independent. While this argument is reasonable, we would expect software vendors to willingly develop to the POSIX API.

In recent years, however, this has not been the case. Developers have not entirely embraced systems standards for application development because the standards typically take three to five years to make it through the IEEE or equivalent standards committee. Developers frequently need new solutions to existing software projects, and cannot wait for the standards committee to devise a standard implementation. This reality is demonstrated by the relatively few POSIX compliant applications being produced or marketed today, several years after the POSIX standard was implemented. This leaves DOD with the reality that most non-developmental items are not POSIX compliant, and are unlikely to be in the near future. Selection of a NOS for DOD information systems must consider this carefully, and evaluate the cost-benefit of adopting a NOS consistent with the market trend to ensure compatibility with the installed pool of non-developmental items.

d. Software reuse

Many IT managers argue software reuse is more of a necessity today, given the increasing complexity and size of software projects. Repositories of software modules, procedural calls, and common code can decrease developmental time significantly, reduce maintenance costs, reduce development schedules, and increase product quality. Many application programs have the same general "look and feel" of toolbars, buttons, and other common functionality. This code can be incorporated in modules that can be reused each time a new application is developed.

A drawback to software reuse, however, is the time required to make that module or software program code reusable. The use of global variables, documentation, and general standardization of code is important and time consuming and may not increase the development efficiency of the current application if it is developed from new program code. While software reuse is a very important concept, it is not really a significant issue that effects the selection of a NOS for use in an information system.

e. Resource sharing

While sharing of file servers, print servers, peripheral devices such as scanners, printers, modems, as well as software and data improves the efficiency of an organization, the TAFIM TRM makes no inference how this tenet improves development efficiency. Resource sharing in a common NOS does not provide any significant improvements in development efficiency, and is left for discussion in other research areas.

3. Analysis of the Unix and Windows NT architectures

The authors chose to combine the analysis of Unix and Windows NT into one section. There are no strong areas to differentiate between the Unix and Windows NT NOS in the area of improved development efficiency. Both NOSs have a common development environment as defined above. Both NOSs conform with the POSIX 1003.1 implementation of open systems. Common development is a matter of application development management to a given set of standards and needs. While the NOS provides the foundation for application development, improving development efficiency is more a function of implementing common development strategies in an open systems environment than it is in establishing the best standards or environment.

The use of non-developmental products like COTS and GOTS offers clear improvements to development efficiency by enabling DOD developers and program managers to allocate development resources to problems where COTS and GOTS solutions do not exist. It is tempting to propose that Windows NT, with its greater installed base of applications (including Windows 3.1, Windows for Workgroups, and Windows 95 compatible applications), offers the greatest benefits to improving development efficiency. However, it fails to consider the broad needs of the users, or the complexity of application development needed to support many of DOD's needs.

The other area of concern is software specific development. There are numerous GOTS applications currently available for the Unix NOS, including the current software running on JMCIS and GCCS. However, there is a preponderance of Windows NT-based COTS software in the market place that gives the Windows NT NOS a clear advantage over the Unix NOS in other areas. With a large COTS pool, developers can reuse code in

their developmental products easily and efficiently. When COTS solutions are not appropriate or effective, GOTS should be considered. An alternative method for specific DOD related development, where COTS and GOTS products are not available, would be to outsource software rather than to develop software within DOD. The important point here is that market place applications are developed and tested by corporations that specialize in and know how to develop software applications. The DOD should not be in the business of developing numerous applications that are already available in the market place. So whether a Windows NT NOS or a Unix NOS is used does not greatly impact improving developmental efficiency. Even though a qualitative determination of which NOS offers more development efficiency was not performed, it is safe to argue that using COTS and GOTS products with either Unix or Windows NT will improve development efficiency.

4. Summary of findings

There is really no NOS which can provide any significant advantage in development efficiency. Improving development efficiency is more a function of establishing a common set of development standards and policies and being consistent in their application across the development of many information systems.

C. IMPROVE PORTABILITY AND SCALABILITY

1. TAFIM's definition of the objective

The portability and scalability of applications will be improved by applying the following principles:

- **Portability.** Applications that implement the model's paradigms will be portable, allowing for movement across heterogeneous computing platforms with minimal or no modifications. With portable applications, implementing activities will be able to upgrade their hardware base as technological improvements occur, with minimal impact on operations.

- **Scalability.** Applications that conform to the model will be configurable, allowing operation on the full spectrum of platforms depending on user requirements. [DISA03, Vol. 2 p. 2-2]

2. Interpretation of objective

Of all the TAFIM objectives, portability, scalability, and interoperability (discussed in the next objective) are arguably the most important and fundamental issues in the development of any DII compliant information system. It would be foolish to deny that other TAFIM objectives were not important, but, they do not have much significance without strong solutions to portability, scalability, and interoperability problems. Portability, scalability, and interoperability problems are almost always solved by technology solutions instead of procedural or managerial solutions. Common standards provide the base upon which portability, scalability, and interoperability are realized.

a. Portability

Portability enables software to be moved or installed to a machine based on a different processor technology or configuration, with as little recoding as possible. Although NOSs, and OSs in general, are often described as either "portable" or "nonportable," portability is not a binary state, but a matter of degree. The crucial question is not whether software will port (most will eventually), but how difficult it is to port. Likewise, portability is affected by both hardware and software compatibilities. At the hardware level, generational differences between processors or differences in

microcoding can significantly affect the porting of a NOS. At the software level, portability can apply to the NOS, applications, data, and data structures.

Developing a NOS that is easy to port is similar to developing any portable program. First, as much of the code as possible must be written in a language that is available on all target platforms. Factors that must considered are the hardware architecture and the microprocessor. Usually this means that code must be written in a high-level language, preferably one that has been standardized. Assembly language code is inherently nonportable.

Second, porting application software or NOSs to obsolete hardware adds complexity to the NOS design. Different hardware imposes different constraints on an OS. For example, a NOS designed for 32 bit addresses could not be ported (except with enormous difficulty) to a machine with 16 bit addresses. As the DOD's needs for information processing expand, outdated computer architectures like the Intel 286 and 386 based personal computers should be phased out of information systems. As our need for information places greater demands on hardware capabilities, NOSs cannot maintain compatibility with legacy systems that do not have the computing power to meet today's information needs.

It is also important to minimize, or eliminate wherever possible, the amount of code that interacts directly with the hardware. Hardware dependencies can take many forms. Some obvious dependencies include directly manipulating registers and other hardware structures or assuming a particular hardware configuration or capacity. This argument ties closely to another TAFIM objective, achieving vendor independence, by

establishing non-proprietary specifications for hardware and software based on *open* standards.

Finally, hardware-dependent code, if any, should be confined to a few easy-to-locate modules. Hardware-dependent code should not be scattered throughout the OS. Instead, it can be hidden in a hardware-dependent structure within a software-defined, abstract data type. Other modules of the system manipulate the data type rather than the hardware by using a set of generic routines. When the OS is ported, only the data type and the generic routines that manipulate it must be modified.

Portability is an essential element in the DII COE. Given the broad demands and needs for information processing, displaying information, and system performance, information systems following the DII COE will be based on several hardware architectures, including desktop personal computers, high performance multi-processor personal computer systems, traditional workstations, and high performing multiprocessor workstations and supercomputers. The NOSs used in this type of information system must be portable across the entire range of systems.

b. Scalability

The traditional definition of scalability has been oriented around software that performed well under a wide variety of usage scenarios. True scalability is about protecting the investment an enterprise makes in its information systems. The real proof of an information system's scalability is its flexibility to adapt to changing needs and its ability to adapt to improving technologies, all without having to be completely replaced or rewritten. The DOD expects and demands that its information systems grow and adapt.

DOD needs flexible, yet cost-effective solutions that can efficiently meet the needs of the users under a wide range of conditions and environments, without consuming excessive resources. Despite this requirement, there are too many projects where information systems do not scale beyond current needs.

Most development teams strive for scalable software to achieve a single body of code that will serve various sizes of installations - for instance, meeting the needs of the one-person office in a remote site as well as the needs of the company's metropolitan corporate headquarters housing thousands of employees. In addition to this restricted view of scalability, there are even more issues to consider.

New technologies have redefined who users are, where they interact with systems, and even how they use corporate applications. System and application design must take into account users at remote sites and mobile users accessing systems while working in the field or at sea.

Another factor emerging today is systems that use multiple architectures, linked together into one large application. This new mix of users requires greater skill and planning on the part of the development team. Given the broad need and wide ranging requirements of DOD information systems using the DII COE framework, it is unrealistic to expect a globally homogeneous hardware architecture.

While the traditional definitions of scalability are important, we are faced today with a need for new definitions and tests for scalability in an ever increasingly connected network environment. In today's dynamic computing environment, it is difficult to imagine any DOD application that does not require the ability to grow and adapt. In

more generic terms, scalability is viewed as the performance gained from resources added to a computer system [INTE95].

The hardware, NOS, application software, and flow of work done on the system together determine the size of the performance gain. While all four of these factors are critical to an information system's scalability, the NOS design is essential to integrate and capitalize on each of the factors to provide overall scaling performance. The NOS must support single processor as well as symmetric multi-processing (SMP), wide ranges of memory addressing, wide ranges of disk storage capacities, and for servers, varied levels of transaction requests. Similarly, the NOS must support distributed computing under a wide range of system loads and application software.

Businesses that employ scalability in their applications will find their systems to be more responsive and flexible in today's dynamic business environment. Software that scales can be reused, whether it is to enable use in another division, or to contend with continuing change and movement in the enterprise. Today, an information system must provide for changes in numbers of users, amounts and types of data, remote and mobile users as well as onsite users, a mix of server types, and lastly, be flexible enough to easily incorporate technological improvements and ever increasing software and hardware capabilities.

In short, building scalable information systems is one of the best ways the enterprise can protect its investment in computer technology. Scalability can be achieved through careful application planning and design, good software development practices, and application of available platforms, programming, and database tools. Figure 11 offers

a simplified example of how hardware and software must scale for the wide range of changing needs of a DII information system.

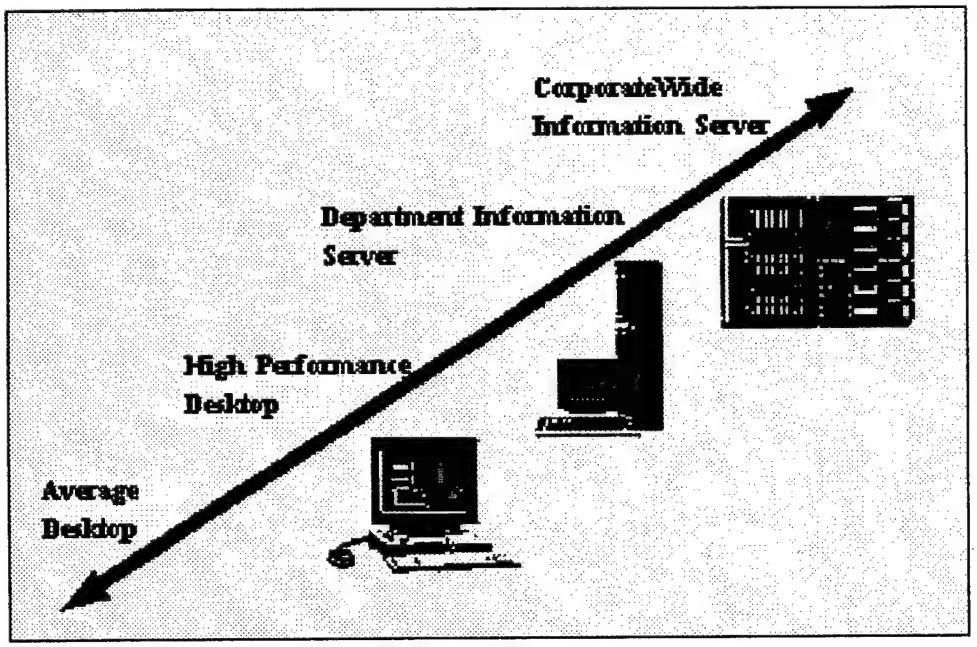

Figure 11. The range of scalable systems

3. Analysis of the Unix architecture

a. Portability

The Unix NOS is generally regarded as the most portable NOS on the market today [DUNP94, p. 467]. Unix portability stems from its strong foundation on *open* systems standards. As a NOS, Unix has been ported to most hardware architectures including those based on RISC microprocessors as well as the Intel 80x86 based CISC microprocessors. Even though there is no disputing that Unix is a very portable NOS, this must be kept in perspective. Unix's reputation for portability is based on two premises. On the macro scale, Unix portability is evidenced by the 15 or so major variants of Unix on the market (see Figure 7). These variants represent the successful porting of the Unix NOS by different vendors. In the context of TAFIM and the DII COE framework,

however, this evidence of portability represents little significance since only two have been adopted for the DII COE.

The true significance of Unix portability lies in the ability of a particular variant of Unix to port to different hardware architectures as needed. In the context of the DII, this is a very necessary feature, enabling the NOS to be used on a wide variety of hardware architectures. Hardware architectures used in most information systems using the DII COE are likely to span the entire spectrum of scalability. This implies that any DOD approved variants of Unix (HP-UX 9.01, or Solaris 4.1.1) must be able to port to hardware architectures being used on the information systems. These architectures are likely to include a wide variety of RISC based architectures as well as the Intel X86 based platforms.

Porting Unix to these architectures must consider the following hardware specific items:

- Data path size - 16 bit or 32 bit
- Byte ordering and byte alignment
- Address size - 16 bit or 32 bit
- Use of microcode
- System bus architecture
- Register operations
- Addressing modes
- Stack management
- Memory management
- I/O Architecture

- Interrupt levels
- Instruction pipelining [ANDL90, p. 218]

Each of these hardware implemented technologies is accessed by the NOS in specific manners. When porting a NOS to architectures which implement these technologies in different manners, the porting process requires adapting the NOS to the specific technology.

Unix was designed using a portable language. This helps the porting process. Additionally, the Unix NOS is based on a modular design. Code which is machine dependent is located in a small number of isolated modules. The modules which contain this machine dependent code are the *machdep* and the *macherr* modules, as well as some isolated assembly coded modules. The *machdep* module contains startup code that identifies the OS, displays real and available memory sizes, and sets up the map from the memory mapping register. It also includes the code to start the system clock, send an interrupt to a process, copy a number of bytes from one physical memory, create a duplicate of a process, change protection codes of text (code), check size of the data, text, or stack of a process, manipulate page tables, and set up initial memory. The second module, *macherr*, has routines to check the CPU state and to process memory parity, CPU time-out, or bus errors.

These two modules, along with the assembly language routines, are required to be rewritten completely for multiprocessor operation. The rewrite becomes more involved if byte ordering and memory management of the target system are significantly different from that of the porting source machine. [ANDL90] While Unix is portable, HP-UX has not been ported to any other hardware platform other than the

Hewlett-Packard line of computer systems. HP-UX will not run on non-Hewlett-Packard hardware architectures. [CUMM96]

b. Scalability

Unix is generally regarded as highly scalable. Unix, running on traditional RISC based workstations such as Sun and Solbourne computers, provides users with unique scalability, supporting massively parallel processing architectures, symmetric multiprocessing, and large capacities of addressable random access memory (RAM) and secondary storage (RAID drives, etc.).

HP-UX, one of the two variants of Unix approved for the DII COE, is designed to run in environments ranging from the average desktop, engineering workstations, workgroups, and departmental servers to enterprise (DOD wide) server systems within the data centers of large enterprises such as in DOD. HP-UX supports SMP with up to 14 RISC-based processors. HP-UX also offers support for enterprise level parallel server support for "parallelized" applications. This feature enables servers to work in parallel to provide information and data retrieval services under heavy user loads. The HP Enterprise Parallel Servers (EPS) comprise two or more HP 9000 T-class or K-class SMP supernodes, each with up to four or 12 SMP processors respectively. Up to 32 supernodes can be configured into a single (EPS) -- yielding a total of 384 processors if 12-processor supernodes are used. This kind of scalability enables the HP-UX NOS to be based on single processor workstations for single user access, as well as on SMP based servers using up to 14 processors, and supernode clusters of HP-UX machines with up to 384 processors.

HP-UX also supports large scale functionality including support for up to 2 billion user IDs, 3.75 Gigabytes of addressable RAM, support for disk striping, a form of virtual disk allocation which spans multiple physical hard disks, and single file sizes of up to 128 Gigabytes. While Unix scales well to the high-end enterprise servers and superservers, it lacks in its ability to provide cost effective low-end scaling. Unix and its associated RISC-based hardware typically costs considerably more than its PC counterpart. Some may argue that this reflects the superior performance and capabilities of the machine architecture and NOS combination, but it is crucial to recognize that it is important for information managers to select the appropriate scale system for each application. We could all accomplish our daily work using a SMP or parallel enterprise server as our desktop machine, but it is not practical for DOD to expend limited financial resources to do so.

4. Analysis of the Windows NT architecture

a. Portability

Windows NT was designed for easy porting and scaling. In fact, portability and scalability are two of the design criteria for the Windows NT development team. While our interpretation of portability clearly states that portability is not a binary state, porting Windows NT can be described as easy to port to a wide range of hardware architectures, based on both CISC and RISC microprocessors. Several design features of the Windows NT NOS permit easy porting, they are:

- **Portable C.** Windows NT is written primarily in the C language, with extensions for Windows NT's structured exception handling architecture. Developers selected C because it is standardized and

because C compilers and software development tools are widely available. In addition to C, small portions of the system were written in C++, including the graphics component of the Windows environment and portions of the networking user interface. Assembly language is used only for parts of the system that must communicate directly with the hardware (the trap handler) and for components that require optimum speed (such as multiple precision integer arithmetic). However, non-portable code is carefully isolated within the components that use it.

- **Processor isolation.** Certain low-level portions of the OS must access processor-dependent data structures and registers. While, the code that does so is contained in small modules, they can be replaced by analogous modules for other processors with relatively little programming effort.

- **Platform isolation.** Windows NT encapsulates platform-dependent code inside a dynamic-link library known as the hardware abstraction layer (HAL). Platform dependencies are those that vary between two vendors' workstations built around the same processor - for example, the MIPS R4000. The HAL abstracts hardware, such as caches and I/O interrupt controllers, with a layer of low-level software so that higher-level code need not change when moving from one platform to another. [CUST93]

Windows NT was written for ease of porting to machines that use 32 bit linear addresses and provide virtual memory capabilities. It can move to other machines as well, but at a greater cost in reprogramming additional modules. As the hardware architecture deviates from standard PC and workstation architectures, the porting process relies more heavily on the modular object oriented design of Windows NT. Similarly, as 64 bit computer architectures become more available for DOD applications, Windows NT will facilitate upgrading and porting to such technologies because modules will be recoded for 64 bit computing. Some of these modules include the I/O manager, Kernel, and HAL (Refer to Figure 9, Windows NT Architecture).

b. Scalability

Until recently, understanding how Windows NT scales in enterprise computing environments has been misunderstood. This misunderstanding has stemmed from confusion over Microsoft's standard Windows NT licensing policy, and a lack of application software products optimized for use with Windows NT.

Microsoft designed Windows NT for use on systems scaling from a single processor to as many as 32 microprocessors. The confusion over this broad scalability is that Microsoft licenses Windows NT Workstation for systems using up to 16 processors, and Windows NT Server for systems using up to 4 processors. Scaling Windows NT beyond the basic license requires special Microsoft Original Equipment Manufacturer (OEM) versions and licenses. While this may sound like a complicated issue, vendors selling multiprocessor computer systems ship Windows NT versions and licenses with support for the number of processors used in multiprocessing systems.

Although we generally think of scalability in terms of the NOS and it's processors, scalability is affected by the applications the NOS runs. A workstation or server, which is scaled to the higher performance end, will typically have numerous processors, vast memory and storage capabilities, and most likely numerous simultaneous users and transactions. While Windows NT can handle this level of scalability, it also requires that applications be programmed to scale and take full advantage of the NOS. One test of this premise compared performance benchmark results of different versions of Microsoft's Structured Query Language (SQL) server on the same system running Windows NT v 3.51. The test results found that the more recent version, version 6.0, performed better. When the hardware consisted of platforms with multi-processors,

version 6.0 performed better, establishing that scalability is a function of both the NOS and the application. [INTE95, p. 3] Figure 12 shows the performance results of the two versions of SQL based on varying number of processors.

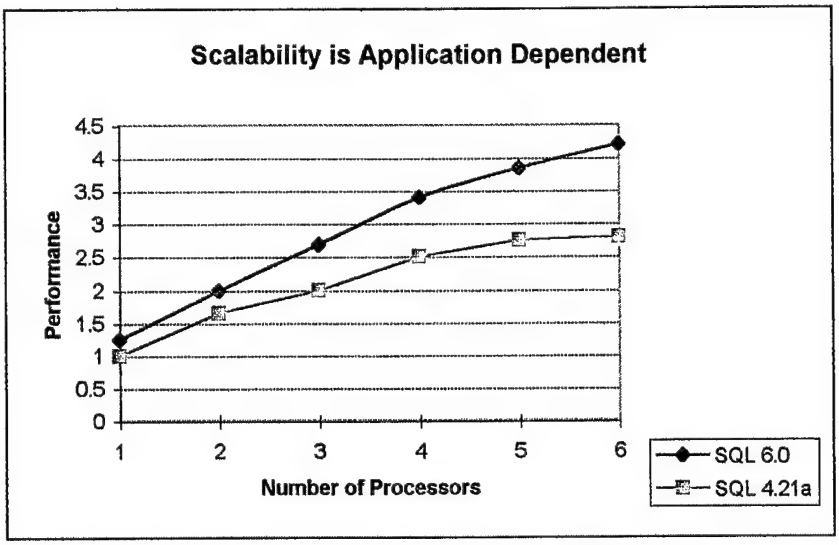

Figure 12. Application Scalability under Windows NT

The number of applications optimized to run on highly scaled Windows NT servers (departmental and enterprise servers with SMPs) is still relatively small. Market analysts forecast that current market trends towards Windows NT as less expensive alternatives to the Unix workstation/NOS will promote application scalability on an already scalable Windows NT NOS. [INTE95] Testing has shown that on large multiprocessor systems running software applications designed for multiprocessing NOSs, that the performance of Windows NT is extremely good, and performs well at the high end enterprise level server environment [INTE95].

5. Summary of findings

Improvements in portability and scalability have been goals that software developers, particularly NOS developers, have been working on for several years. Even

though we regard Unix as a very portable and scalable NOS, vendor specific implementations of Unix lack the portability across different hardware architectures. Windows NT offers a wider range of porting options to a wider base of hardware architectures.

Information managers should find the NOS which scales best to its application mix. Windows NT scales best for low end single user workstations to the mid-level or departmental servers (supporting both single processor and SMP), while Unix scales best from the mid-level servers to the high end servers. This "scale to fit" concept for selecting the appropriate NOS then requires that the NOS(s) be interoperable, and is the focus of the next section.

D. IMPROVE INTEROPERABILITY

1. TAFIM's definition of the objective

Interoperability improvements across applications, hardware, and mission areas can be realized by applying the following principles:

- **Common Infrastructure.** The DOD will develop and implement a communications and computing infrastructure based on open systems transparency including, but not limited to, operating systems, database management, data interchange, network services, network management, and user interfaces.

- **Standardization.** By implementing standards from the DOD Profile of Standards, applications will be provided and will be able to use a common set of services that improve the opportunities for interoperability. The standards provided in TAFIM are included in appendix B (Table 4). [DISA03, Vol. 2 p. 2-3]

2. Interpretation of the objective

As mentioned in the previous objective, interoperability is one of the most important objectives being analyzed. If information systems can not interoperate, then many of the objectives of the TAFIM TRM are irrelevant. An information system does not consist of stand alone desktop computers working separately, but a strongly cohesive, highly coupled network of compatible and interoperable machines. Computers operating in isolation do not provide significant informational resources in an organization, but when interconnected, they provide users with a synergistic collection of information which may reside anywhere on the network.

a. Common infrastructure

Development of a common infrastructure enables a series of vendor independent hardware and software the ability to communicate and run applications anywhere on the network. A common NOS, common database, and network implementation are all areas to be considered in achieving this common infrastructure. Open systems, defined in Chapter II and discussed throughout this thesis, provides one of the essential mechanisms to achieve this common infrastructure and interoperability. A common infrastructure includes such hardware and software as the NOS, databases, communications protocols, as well as the network and its interface. A common infrastructure is not about a specific NOS or communications method, but a common or shared ability to communicate with the infrastructure. A common infrastructure should provide systems that are able to communicate with not only existing systems, but are also capable of being integrated into future systems.

There are many forms of integration. At a minimum, integration of Windows-based and Unix systems must provide for simple network connectivity between the systems. Users must be able to access files and data across platforms over a network, and applications on different systems must be able to communicate with each other. To achieve better integration, it is also necessary to enable cross-platform application development, object services, database access, messaging, and systems management. See Figure 13, Interoperability between PCs and Unix Workstations.

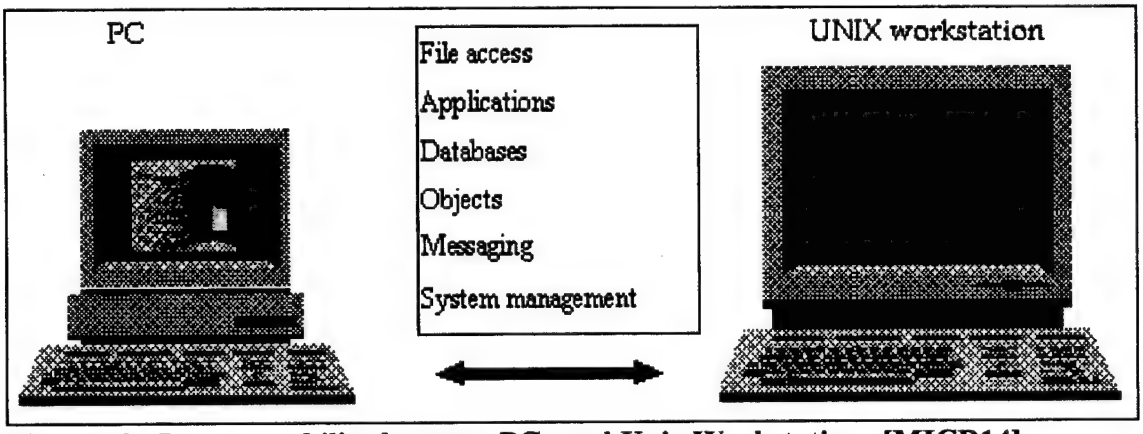

Figure 13. Interoperability between PCs and Unix Workstations [MICR14]

With cross-platform application development based on standards, developers will be able to write platform-independent applications, and then tailor the application to its appropriate scale. Cross-platform object services enable software components to communicate across platforms easily and can help make users more productive. Object services, database access, and messaging provide similar advantages. System administrators will be able to manage heterogeneous systems, if system management software can provide, at one place, management information about these heterogeneous systems running different NOSs. Cross-platform database and messaging

services provide users with a means for easy, platform-independent information exchange. [MICR14]

At a minimum, system integrators must arrange to provide users with a means of accessing files between systems of different NOSs. One way of achieving this interoperability at the NOS level is to enable dynamic loading of installable file systems when the NOS encounters a non-native file system. The concept of the installable file system (IFS) is to permit the NOS to load the appropriate device drivers and interpreters to provide the NOS access to the file system (e.g., FAT, NTFS, NFS). Each file system stores data in different methods, requiring the NOS to be aware of the storage methods through the use of the IFS drivers. IFS drivers can be loaded during system boot, or dynamically as needed.

The NOS provides the connectivity interface between the machines, LANs and WANs. Future DII information systems will likely be a heterogeneous mix of vendor independent platforms conforming to a common infrastructure. It is this common infrastructure and carefully selected NOSs that must support interoperability.

b. Standardization

The problem of non-compatible information systems can only be accomplished through standardization. For systems to interact, there needs to be an accepted set of standards not just within organizations, but globally. If individual organizations develop and implement their own set of standards, then there will be inconsistencies between these organizations. One might argue that it is impossible to adopt a global series of standards because new and improved standards are being

developed in the market every day. This thesis argues that specific standards adopted for an information system are less important than the market agreeing upon an acceptable set of standards that provide worldwide (or global) interoperability.

Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose. For example, the format of credit cards, phone cards, and "smart" cards that have become commonplace is derived from an ISO standard. Adhering to the standard, which defines such features as an optimal thickness, means that the cards can be used worldwide. International standards thus contribute to making life simpler, and to increasing the reliability and effectiveness of the goods and services we use, wherever we use them. [ISOR96]

While to some extent it does not matter which standards are selected, some offer more and better functionality than others. There are numerous proprietary standards, as well as multiple *open* systems standards. For example, DOD information systems are based on *open* systems standards because they support objectives such as vendor independence and development efficiency. While the DOD strives to achieve these objectives, the market is developing and adopting its own set of standards. We argue that the DOD should not be in the business of trying to develop or regulate standards, but allow the market to decide on the standards and then adopt them whenever they fit DOD's needs. Computer industries are developing new technologies every day because of their competitive nature, and these new technologies should be utilized in DOD.

Additionally, once standards are developed in the market place and approved by international committees, they should be implemented in DOD information systems. Implementing standards has the following benefits for the computer industry:

- Increase market access and acceptance
- Reduce time and costs in product development
- Attain a competitive advantage and faster time to market
- Cut costs in component and materials acquisition
- Reduce administrative and material expenses [ANSI96]

While standards are clearly needed to achieve interoperability across the platforms of an information system, some standards evolve differently than others. This evolution can have dramatic impacts on DOD information system development initiatives, and explains why DOD has not divorced itself from developing its own standards like TCP/IP.

There are essentially two types of standards in the market: *open* systems standards and proprietary. *Open* systems standards are developed by consensus and typically take about three to five years to make it through very large committees such as the IEEE. This length of time can mean that once a new technology becomes an IEEE standard, it may have been in use for several years, or may even be an old technology. Proprietary standards, on the other hand, are typically developed quicker by vendors who have a stake in the success of their product on the market. Proprietary standards are in conflict with some of the other TAFIM objectives, such as vendor independence.

The benefits of standards are in line with the new role of DOD, which is "to do more with less". It is evident, therefore, that DOD should implement commercially

developed and internationally approved standards to achieve compatibility and interoperability.

3. Analysis of the Unix architecture

a. Common infrastructure

Unix achieves a communications and computing infrastructure based on open systems transparency, by providing flexible and open alternatives, rather than being locked into proprietary systems and applications. Although there are many proprietary variants of Unix, it is still arguably the most *open* operating system on the market. A common infrastructure of vendor independent hardware and software puts the control of information technology decision making into the hands of organizations rather than into the hands of specific vendors. Organizations are weary about giving up control of their information systems to specific vendors. With an *open* NOS, organizations can adapt hardware, applications, and other software to different or better versions as they arise, and not wait for a specific upgrade from a specific vendor. Unix achieves the advantages of cooperative *open* systems development with the advantages of a freely competitive market. [UNIF95]

According to UniForum, the International Association of Open Systems Professionals, Unix has been cooperatively developing information systems for the past 25 years. Many of the mainframes and legacy systems of the past are being converted into Unix based open architectural information systems. Without a single vendor to rely on, modification of systems to meet current needs can occur with relatively little capital investment. Portability of applications is relatively easy and can be accomplished with

modest cost. Vendors come and go, but with *open* systems, Unix provides a stable and secure platform.

b. Standardization

As stated above, Unix vendors such as HP and AT&T have been developing products for many years. As a result, they have established some of the primary industry standards readily available and in use today. These standards have helped to shape information system interoperability. Network interoperability use to be impossible unless everything was bought from a single vendor. Unix, on the other hand, was the first OS to provide communication on the Ethernet with TCP/IP. TCP/IP was designed and adopted as the Internet communications protocol suite in 1983. [UNIF95]

One major standard from the Unix community is the X-Window System. The X-Window system provides the ability for other OSs to interact with it. X-Windows is a sophisticated windowing system developed and overseen by a nonprofit, vendor-neutral consortium. It is therefore relatively easy to connect diverse laptops, terminals, and desktop computers to Unix servers, making network-wide interoperability a reality.

Other standards from the Unix community include Simple Network Management Protocol (SNMP) and Simple Mail Transport Protocol (SMTP), which are part of the TCP/IP suite of protocols. SNMP is a simple mechanism to centrally manage diverse networks and will be discussed in detail later in system manageability. Additionally, the development of the programming languages C and C++, now standard languages used in most OSs, helps programmers more easily design NOSs with interoperability in mind.

4. Analysis of the Windows NT architecture

a. Common infrastructure

Windows NT is a NOS which was first introduced into the commercial market place in 1991. Since then, Windows NT has undergone two major upgrades. Windows NT was not the first NOS designed to exist on both local area and wide area networks, but it was, unlike other NOSs, built from the ground up with wide connectivity and interoperability in mind. Windows NT provides:

- Portability across families of processors, such as the Intel 80X86 and Pentium lines
- Portability across different processor architectures, such as CISC and RISC
- Transparent support for single-processor and multiprocessor computers
- Support for distributed computing
- Standards compliance, such as POSIX
- Certifiable security, such as C2 and F-C2, E3

Windows NT is a complete NOS with fully integrated networking, including built-in support for multiple network protocols. These capabilities differentiate it from other OSs and NOSs such as DOS, Windows V3.1, and Unix. With these OSs, network capabilities are either installed separately from the core operating system, as an after market add-on, or patched in a version upgrade.

Windows NT offers built-in support for both peer-to-peer and client/server networking. It does not provide host based networking. Windows NT provides interoperability with, and remote dial-in access to existing networks, support for distributed applications, file and print sharing, and the ability to easily add networking

software and hardware. Windows NT does this by design, closely following the OSI reference model. The Windows NT implementation of the OSI reference model is displayed in Figure 14. Windows NT uses this model to provide services to the next higher layer, shielding the higher layer from the details of how services are actually implemented. In Windows NT, network layers provide virtual communication with peer layers on another computer. In reality, each layer communicates only with adjacent layers on the same computer.

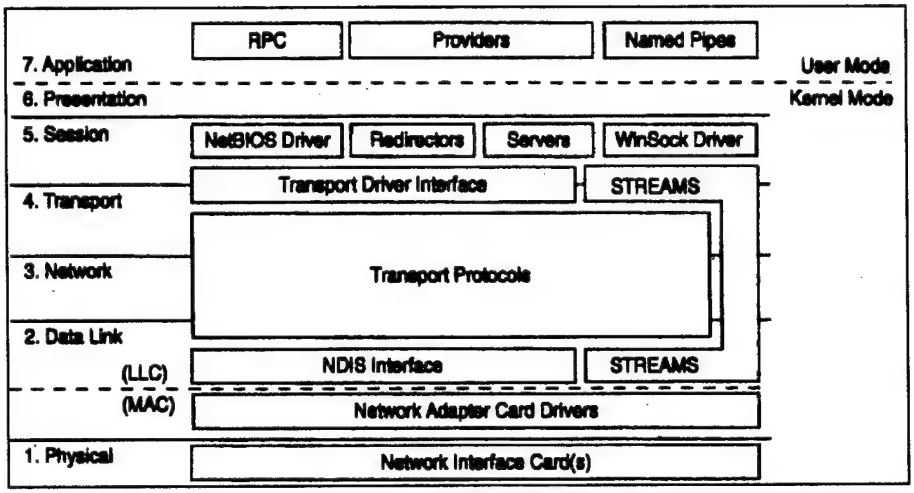

Figure 14. NT implementation of OSI Reference Model [MICR03]

b. Standardization

Adoption of the Industry Standards Organization (ISO) model in the Windows NT design offers Windows NT a standards-based method of interconnection, and thus, increases interoperability with other machines using the same standardized model. It also allows NT to connect with non-conforming NOSs due to the nature of the layered design, providing services with the adjacent layer of the local machine. [MICR03, p. 14]

While the built-in networking features of Windows NT are quite sophisticated, they suffer from being proprietary and lend themselves to providing excellent interoperability with other Microsoft products like Windows 95, Windows for Workgroups, and LAN Manager. Providing interoperability with other OSs like Novell NetWare, IBM's OS/2, Macintosh, and Unix, requires the Windows NT architecture to provide an alternative mechanism for connectivity. Microsoft provides this at two levels: the Network Device Interface Specification (NDIS) and the Transport Driver Interface (TDI).

NDIS provides an interface for communication between the Medium Access Control (MAC) sublayer and protocol drivers higher in the OSI model. This standard is key to isolating the details of the Network Interface Card (NIC) from the transport protocols and vice versa, and eliminates the need to write complicated device drivers for each type and brand of NIC. TDI provides a direct link between all redirectors or network file systems and other network transport drivers. Since Windows NT sees all networks as some type of file system, network providers need only provide program code for their file system, which Windows NT loads as required to access the file system of that network. This code is called an Installable File System (IFS). This concept is similar to the "IFSHLP.SYS" file installed from the "CONFIG.SYS" file in Windows for Workgroups to give 32 bit disk and file access.

With TDI and NDIS, Windows NT can access many network file systems by adding the installable file system code and the transport driver for the type of networking protocol that is in use. Table 2 shows the protocol support that Windows NT provides:

Network Operating System	Protocols used
Windows, WFW, Windows 95, Windows NT	NetBEUI
Unix	TCP/IP
OS/2	NetBIOS
NetWare	IPX/SPX
Apple	Appletalk
IBM Mainframes	DLC

Table 2. NOS Protocols [MICR09]

While other NOSs support only their own proprietary protocols, Windows NT is designed to interoperate, and therefore, can simultaneously support all of the above listed protocols. Multiple protocol support is achieved by maintaining multiple stacks in memory. The NOS then redirects incoming network packets to the appropriate protocol stack for processing and routing. This provides Windows NT with a significant ability to integrate heterogeneous networks into a single network running multiple protocols.

Windows NT, like Unix, uses the Remote Procedure Call (RPC) facility. The RPC is highly used in distributed computing and is the IPC method of choice for software developers. Windows NT is fully compatible with the standardized RPC specification. The RPC facility in Windows NT is powerful because it relies on other IPC mechanisms to transfer function calls between client and server. This way, Windows NT RPCs can use named pipes (Unix), NetBIOS (OS/2), Windows Sockets (WFW, LAN Manager), etc. to communicate with remote systems. Windows NT's IPC flexibility makes the RPC feature one of the most flexible, portable, and interoperable IPC mechanism of the leading NOSs. [CUST93, p. 315]

5. Summary of findings

In the final analysis, Unix and Windows NT are both very interoperable NOSs. Unix traditionally has been the platform of choice in engineering and scientific computing, and has established a long standing tradition as a stable and efficient NOS. Windows NT, on the other hand, has demonstrated its ability to offer features, designs, and benefits which exceed the abilities of many others.

In general, Unix excels in the areas of networking, communications among heterogeneous systems, and processing-intensive applications. Unix is mature and has been the replacement NOS as companies have replaced mainframes with workstations over the past 20 years. On the other hand, Windows NT combines impressive interoperability with the familiar Windows GUI. Some organizations will choose to have both NOSs as part of their information systems, with Unix performing at the high end server and Windows NT at the desktop and midrange and low end servers. However, these organizations should recognize the resource costs involved. It is desirable then, that these two NOSs be interoperable. As a result, we briefly discuss interoperability between Unix and Windows NT.

Microsoft designed Windows NT to be interoperable with Unix. Having enjoyed much success in providing replacements to mainframes, Unix vendors are not eager to provide integration between Windows NT and Unix. The majority of the Unix community sees Windows NT as a threat rather than an opportunity. [UNIF95]

Some companies have decided to migrate applications to Windows NT in order to use to use less expensive server platforms; nevertheless, they still want to retain their Unix

development environment. As a result, the most typical environment in which NOSs coexistence is observed today is a three-tier client/server architecture. A three-tier architecture includes the high end servers, midrange servers, and clients. Many companies have reengineered their information technology to support distributed computing, including a back-end server typically running Unix; a mid-level server for file and print services that uses Unix, Windows NT, or another NOS; and PC clients running a Microsoft desktop NOS. It is, therefore, in the interest of some companies to have these two NOSs interoperate and provide maximum services and flexibility to users.

It is recognized that companies that adopt information systems using two or more NOSs will see an increase in life-cycle costs, training costs, system management costs, etc. While a two NOS information system is more expensive to maintain, it offers the benefit of permitting NOSs to be used where they perform best (e.g., Windows NT on desktop machines and departmental servers, and Unix on enterprise servers). Until a single NOS becomes available which completely captures the full range of functionality from desktop computers to enterprise servers, DOD must consider the cost benefit of adopting more than one NOS.

E. PROMOTE VENDOR INDEPENDENCE

1. TAFIM's definition of the objective

TAFIM states that vendor independence will be promoted by applying the following principles:

- **Interchangeable Components.** Hardware and software supporting or migrating to open systems compliance will be acquired or implemented,

so that upgrades or the insertion of new products will result in minimal disruption to the user's environment.

- **Non-Proprietary Specifications.** Capabilities will be defined in terms of non-proprietary specifications that support full and open competition and are available to any vendor for use in developing commercial products. [DISA03, Vol. 2 p. 2-3]

2. Interpretation of objective

TAFIM uses the term "Promote Vendor Independence" as the title for this objective and states how to achieve it, but does not provide a definition for it [DISA03, p. 2-18]. Vendor independence is often thought to be one of the results of achieving *open* systems. Vendor independence is a benefit to the DOD. With the ability to choose computer components, both hardware and software, from a variety of vendors, the DOD has greater flexibility. This greater flexibility promotes competition which in turn can keep costs down. No longer is the DOD locked into purchasing components from one proprietary vendor.

The idea is that if the industry followed a specific set of standards for every piece of software built, then the DOD and other consumers could gain vendor independence. For example, if Microsoft Word and Corel WordPerfect both followed a standard for interface commands, styles, etc., then system administrators could swap products and the users would feel minimal impact. This concept of vendor independence in software can be compared to SCSI hard drives of today. Computer users can buy a SCSI hard drive from a variety of manufacturers, all with the same connectors, physical dimensions, etc. This compatibility allows the decision of which manufacturer to purchase from to be based more on slight performance enhancements, price, storage size, and guarantees, rather than only being able to purchase from the company that manufactured the computer.

a. Interchangeable components

Having interchangeable components means that the DOD has the ability to purchase computer components from a variety of hardware and software vendors, as well as the capability to have these components function together in one system. Independence from vendors is achieved by developing standards that define the way components from different manufacturers interact with each other and the NOS. This ability to have interchangeable hardware and software components provides several benefits. Benefits include lower costs, a lower investment risk, greater flexibility, and greater scalability. All of these benefits provide increased economic advantages to the DOD.

Interchangeable software components require software be portable and cross-platform compatible. As discussed earlier in this chapter, portability is essential to any DII COE information system and lends itself to supporting independence from vendors. This benefits the DOD. One benefit, lower costs, results from the ability to port applications from one NOS to another, due to increased competition. With portability, programs written on one computer system would be able to run on another computer system unchanged, merely recompiled. Both time and money are saved since production, distribution and training costs are reduced because the number of duplicate applications is reduced.

Interchangeable components also result in reduced investment risk. If components are made to be interchangeable, organizations reduce their exposure to uncertain markets because new software and hardware is developed according to published *open* systems standards. No longer will users be locked into particular hardware and software solutions. Because of standardization efforts, there is little pressure to make

a large initial investment in new technology, which might not be mature. Instead, organizations will be able to scale or upgrade their *open* systems according to their own needs, on their own schedule. This independence from vendors will enable organizations to increase their leverage of technical resources, by not requiring them to rely on proprietary vendors.

Independence from vendors has been pushed recently in the market. One recent implementation of this concept is called "plug and play." "Plug and play" is a notion of hardware and software components working together like black boxes to perform a task. The user is not concerned with how the components perform the task, only that the task is completed in the same consistent manner, regardless of the manufacturer. If the box breaks, the user can simply go to any vendor and purchase a replacement that will plug into his system and perform the same task.

In order for "plug and play" to work, the NOS must be able to support it. This requires that the NOS be capable of accomplishing several tasks. The NOS needs to be able to communicate with hardware, use generic drivers, and set Interrupt Request lines (IRQs) and High Memory Addresses. With "Plug and Play," the NOS is able to perform these functions transparent to the user.

NOSs need not only support interchangeable components, but they need to be interchangeable themselves. This means that a NOS needs to function on a variety of hardware types (e.g., RISC, CISC, etc.). The user should be able to remove his current NOS and install a new one, maintaining functionality of both hardware and software.

In order to accomplish these tenets, the NOS must be platform independent. Therefore, it must be written in a compilable third or fourth generation

language (3GL or 4GL), not machine language or assembly code. This was the intention behind the initial variant of UNIX. Originally, Unix was written in B (a predecessor to C, a 3GL), not in assembly language, so that the author could port it to different platforms easily. This idea of writing a NOS in a generic 3GL allowed three benefits:

- A compiler could take this "portable" code and compile it for the machine in question.
- It was easily changeable and adapted to specific needs.
- It could be implemented on smaller computers (Mini and Micro computers). [ARNO93, p. 2]

The second benefit resulted in hurting the commercial viability of Unix as much as it helped the educational communities. As each user adapted his variant of Unix to fill his needs, it became incompatible with another user's adaptation. With all the modifications made to Unix, it became difficult to develop an application that would operate on more than one variant of Unix. This lack of portability has come full circle over the years as we now see the Unix community trying to move back towards NOS and applications portability.

b. Non-proprietary specifications

Non-proprietary standards are well publicized and available to any vendor. They evolve through industry consensus, and are freely available so that vendors can implement them to develop products that compete in the market place. There are numerous non-proprietary standards in existence today.

Non-proprietary specifications take standards one step further. Standards apply to a group while specifications apply to an instance. A good example of a non-

proprietary specification is TCP/IP. Originally developed by the DOD, any manufacturer or software developer can make products using these protocols and not pay a licensing fee or worry about getting sued.

Proprietary standards are developed by a specific vendor. Their future development is controlled by a single company. There are also many examples of proprietary standards. Examples of proprietary protocols include: AppleTalk, SNA, and SPX/IPX. Each of these protocols was created by one vendor and that vendor has maintained control over the migration and changes for that protocol. Typically, other vendors must pay a licensing fee to implement a vendor's proprietary standard. This type of standard is usually well defined and supports very good interoperability with other computers that use the same protocol.

Non-proprietary standards and specifications are good for the DOD because they prevent the DOD from becoming vendor dependent. Procurement rules limit the language used in contracts by the DOD, often preventing language that yields vendor specific contracts. A non-proprietary standard allows multiple vendors to try their hand at delivering solutions to problems that the DOD needs to solve. By having standards open, vendors do not have to follow a vendor-specific standard. Following proprietary standards requires third party vendors to acquire a license from the standards' owner. Even if a proprietary standard was followed, and the vendor published the standard and charged nothing to license it, the perception would still be that the proprietary standard owner had an unfair advantage.

3. Analysis of the Unix architecture

Unix promotes vendor independence in a variety of ways. Because variants of the Unix NOS are freely available and readily adaptable, many vendors market versions of Unix. Vendors have traditionally taken the Unix kernel and added their own proprietary features to it. These features optimize their Unix variant to fill the market's needs. Several variants of the Unix NOS (e.g., LINUX and Berkeley Software Design Inc. (BSDI)) are *open* enough to allow for easy kernel modification or security enhancements [HUDG96]. For the most part, Unix variants are packaged with particular hardware, so they are closely tied to a single hardware platform.

a. Interchangeable components

The Unix NOS has long embodied the belief of interchangeable components, and thus Unix workstations have always supported it. An example of this are the SCSI peripheral devices which do not require the manipulation of IRQs that are required on Integrated Drive Electronics (IDE) and Enhanced Integrated Drive Electronics (EIDE) drives of WINTEL computers. SCSI is the dominant Unix input/output (I/O) interface standard, but many Unix vendors support other interfaces. Parallel and serial interface standards, more often associated with (e.g., RS-232 Serial Interface) are also supported by Unix. Even standard Enhance Industry Standard Architecture (EISA) expansion slots are supported by most Unix workstations.

Unix NOS itself is interchangeable to a great degree. Because the NOS is written in C, it can be easily recompiled for different platforms. It is important to note, however, that the vendor must do this recompilation because the code usually belongs to

them. As mentioned earlier, LINUX and BSDI are exceptions to this and allow the end user to make necessary modifications, before the end user compiles the code. Unix code is currently available for many platforms, both RISC and CISC, including traditional workstations, PCs, and Macintoshes.

b. Non-proprietary specifications

Unix has a reputation in the computer industry of being an *open* operating system. This is largely due to the history of Unix. As previously discussed, after being developed, Unix source code was given away to universities by AT&T. Because Unix has been available for more than 20 years in an open format, most of the protocols and standards that it relies upon are open. Different publishers of Unix NOSs have, over the years, included their own proprietary changes to the NOS to tailor its functionality. The problem is that with each change, the NOS is less compatible with other Unix NOS variants. The industry now realizes that this hurts Unix as a whole because software manufacturers must spend extra effort to make their products work on all the different variants. There is now an industry wide push toward interoperability, which has led to several *open* systems standards, as discussed earlier.

The Unix NOS also provides an additional benefit in that it integrates the same *open* systems network protocol that is used on the Internet: TCP/IP. Unix workstations talk to each other via TCP/IP, and even their printers have IP addresses. This has led to the easy adaptation of Unix into the Internet and secured its position as the leading Internet server.

Unix does support POSIX 1003.1 and 1003.2. However, these standards do not cover all the APIs necessary to provide a user-friendly GUI. As discussed earlier, a GUI is necessary in today's fast-paced, limited training time, military environment. Motif is a style guide, while X-Windows and the Common Desktop Environment (CDE) are vehicles, designed to this deficiency in POSIX.

Motif is based on the X-Consortium's X-Windowing System [MOTI96]. "Motif is a widely-accepted set of user interface guidelines developed by the Open Software Foundation (OSF) around 1989 which specifies how an X-Window System application should 'look and feel'" [MFAQ96]. Motif, the de facto GUI standard for Unix workstations, includes:

- A window manager client called MWM
- A user interface style guide (published by Prentice-Hall)
- C-language programming libraries to help programmers develop Motif-compliant applications
- A user interface scripting language (UIL) that programmers can (optionally) use to specify their user interfaces [MOTI96]

Although the most recent version of Motif is version 2.0, most UNIX workstation vendors are currently using older versions, either 1.2.4 or 1.2.5. This is primarily due to the fact that version 2.0 was introduced recently. [MOTI96] Hewlett-Packard's TAC-4 computers are shipping X-Window system Version 11 Release 5 (X11R5) that is based on Motif 1.2.5 [HEWP95, p. 4].

Version 1 of the Common Desktop Environment (CDE) is based on Motif 1.2.5. The CDE was an effort by Unix vendors such as Sun, HP, IBM, and Novell to standardize on a consistent GUI appearance. This effort has paid off with these and other

companies shipping CDE with the latest version of their OSs. CDE goes one step further than just appearance of the GUI by giving software development tools to software developers as well as a controlled set of APIs. The development of CDE was recently taken over by the OSF. They have subcontracted development work to the X Consortium. [DTKS96]

Unix embraces many *open* systems standards. TCP/IP, Motif, and POSIX are among these standards, but this certainly does not comprise a complete list. The standards discussed are merely some of the more important examples. Nonetheless, they serve to demonstrate the Unix industry's commitment to adopting *open* systems standards.

4. Analysis of the Windows NT architecture

a. Interchangeable components

The PC revolution has grown up around the idea that inexpensive, easily interchangeable components will proliferate computers onto the desktop. By embracing the concept that a solution which is inexpensive but effective is more likely to survive than a concept that is higher quality but more expensive, the PC market has flourished. [GANC95, p. 7] Due to this philosophy, NOSs which are designed for PCs recognize a large number of different hardware products from a large assortment of vendors. The Windows NT NOS is a prime example of this. The version of Windows NT compiled for a PC compatible computer can recognize different busses, peripherals, protocols, and designs (e.g., SCSI, IDE, E-IDE, Micro-Channel, VESA, ISA and PCI).

This provides a tremendous amount of vendor independence. Users have the capability to purchase hardware components from a variety of vendors to run on their

PCs (as long as the hardware works together within the PC) with the confidence that their new hardware will be supported by Windows NT. The wide variety of hardware supported also enables users to take advantage of using older components to allow migration to newer systems without having to purchase new hardware. This allows users to upgrade their systems based on their needs.

Windows NT is also designed to be portable and platform independent. This reduces the need for a consumer to rely on one vendor for a particular hardware solution. Window NT was designed to run on both the Intel x86 CISC family and RISC based processors. It supports a wide range of processors including: 32 bit x86 micro-processors, Intel Pentium, PowerPC, MIPS, R4000, and Digital Alpha AXP. Supporting a wide range of hardware platforms also enables the DOD to migrate to Windows NT without having to invest in a new hardware infrastructure.

There are, however, some areas of concern here. Windows NT is primarily written in C and therefore can be recompiled on different systems with a system specific compiler. However, Microsoft felt that the C programming language was not optimized well for some performance intensive tasks. Microsoft felt that the performance penalty was large enough to warrant rewriting some of the code in assembly language. This portion of the code must be rewritten for every platform that NT is ported to. [RULE95, p. 9]

Additionally, application programs which are written for NT on an Intel processor will not run on a MIPS machine. This is because 3GL languages are compiled into machine language and machine language is platform dependent. The code for the application program must be recompiled for each new platform. DOD organizations need

to keep this in mind when they purchase new hardware running Windows NT and expect to be able to run their existing software.

b. Non-proprietary specifications

Although Windows NT is proprietary, it does support some non-proprietary *open* systems standards. One example of this is the Posix sub-system contained in NT. This Posix subsystem contains the entire API defined by IEEE's 1003.1. Support is limited when it comes to IEEE 1003.2, the portion of POSIX which provides a command-line interface standard as well as certain utilities (e.g., the vi text editor). [BARA93, p. 142]

Another *open* systems standard included in Windows NT is TCP/IP. The use of TCP/IP is built into the NOS and allows connections to the Internet or to Intranets. However, in Windows NT environments, TCP/IP can only interact with other TCP/IP systems via low-level functions like FTP and ping. This is because Windows NT fails to support all of the TCP/IP protocol stack. For example, it fails to support Routing Information Protocol (RIP), a protocol used by routers to communicate with each other. This lack of support makes it difficult for Windows NT to communicate in some TCP/IP environments. [RULE95, p. 327]

5. Summary of findings

Clearly, TAFIM stresses in this objective that vendor independence is achievable by moving towards *open* systems. To that end, there must be compliance by the NOS to some standards. In comparing the two operating systems, Unix has a rich history of supporting open standards. Although NT does support some open standards like POSIX,

its adherence to the POSIX standard is incomplete. This causes a lack of support for some command-line applications that are contained in POSIX. Due to Windows NT's graphical nature, the lack of complete command-line support has little impact on Windows NT applications.

POSIX does not currently support a GUI, and so a solely POSIX compliant application must rely on a command-line user interface. This is one inherent weakness in POSIX, especially in today's graphical computer world. The DII COE includes a standard for a GUI, the MOTIF style guide using X-Windows, but NT is not compatible with these standards.

Windows NT has its own proprietary Win32 API. This API is necessary to operate an application in the GUI NT environment. The Win32 API is owned and controlled solely by Microsoft and therefore is are not an *open* standard. In order for developers to create programs for a Windows NT environment that is POSIX compliant, they must have both POSIX APIs and Win32 APIs. In other words, developers cannot develop a complete *open* systems program on Windows NT. They must use Microsoft's Win32 APIs. *Open* standards are not controlled by a single company. *Open* systems is, however, a major focus of the DII COE.

The Common Desktop Environment (CDE) initiative is a strong move by Unix vendors toward *open* systems standards. In addition to a common GUI, the CDE allows programs to be developed for multiple platforms using a GUI, not just a command-line interface. This capability, combined with the POSIX APIs, allows complete program development.

Both NOSs support interchangeable components. Drivers for hardware have traditionally been supplied by the hardware vendor, but the trend is toward including generic drivers in the NOS. This will complicate the process of portability of the NOS to different machines, because drivers must be written for all hardware types. Also, they are not usually covered by a set of standard APIs, and therefore must be written in assembly or machine language. This could be an exhaustive task which must be repeated each time the NOS is ported to a different platform, inhibiting easy portability.

Given that both NOSs are written in C, theoretically both provide the same degree of portability. As stated, both NOSs would need to be recompiled for the specific hardware they were to run on. The availability of Unix over the years has resulted in Unix being recompiled on many machine types. Windows NT has already been ported to different machine types.

When examining the two principles that TAFIM believes will promote vendor independence, it is evident that both NOSs support a fair degree of interchangeable components. However, Unix supports a wider range of non-proprietary specifications. While it is apparent that Microsoft partially subscribes to the ideals of *open* systems in developing a NOS that provides some degree of portability, scalability, and multi-platform capabilities, the standards that they rely on are truly proprietary. This results in a reliance by programmers and developers on Microsoft.

F. REDUCE LIFE CYCLE COSTS

1. TAFIM's definition of the objective

TAFIM's definition of reducing life cycle costs will be realized by applying the following principles:

- **Reduced Duplication.** Replacement of "stovepipe" systems and "islands of automation" with interconnected open systems, which can share data and other resources, will dramatically reduce overlapping functionality, data duplication, and unneeded redundancy.

- **Reduced Software Maintenance Costs.** Software complexity may increase with increased user demand for services such as distributed processing and distributed database services. However, if the principles described above are implemented, reductions in software maintenance will be realized because there will be less software to maintain. In those cases where the number of DOD users is small, increased use of standard non-developmental software will further reduce costs since vendors of such software distribute their product maintenance costs across a much larger user base.

- **Reduced Training Costs.** A reduction in training costs will be realized because users rotating to new organizations will already be familiar with the common systems and consistent Human Computer Interfaces (HCI). [DISA03, Vol. 2 p. 2-3]

2. Interpretation of objective

The reduction of life cycle costs is a major concern in today's downsizing environment. The accomplishment of reducing costs over the life cycle of an information system relies more heavily on the maintenance costs than the initial acquisition cost. TAFIM outlines three ways to accomplish this goal.

a. Data duplication

The first method that TAFIM describes that will contribute to reducing life cycle costs is removing data duplication. Traditionally, the DOD has developed information systems that have resulted in overlapping functionality, and widespread duplication. NOSs need to be designed with the tools to promote interoperability between systems and data sharing across applications. With these tools, data duplication will be reduced.

An example of widespread data duplication today is the current structure of databases in the DOD. Databases for the DOD are maintained in multiple locations and often contain overlapping information. The compilation of necessary data located in different databases is difficult. When multiple OSs are in use, data sharing is compounded. The DOD needs OSs which promote ease of data sharing and interchangeability so that it can eliminate data duplication.

The widespread overlapping functionality is partially due to the lack of connectivity of computer systems. If all information systems could communicate via standard protocols, then databases could reside at multiple locations and be called upon for information as necessary. Other database management problems such as data format, structure, relational verses flat file format, etc., need to be solved also. A NOS with *open* standard addressing data format (e.g., ODBC), standardized SQL queries, and file formats would allow development of interconnecting databases. The long-standing problem of data and software duplication is something that continues to plague information technology interconnectivity. Global access to information through a common infrastructure, like the DII COE, is intended to help reduce data duplication.

A NOS can also promote eliminating data duplication by allowing applications to interoperate to perform common tasks. A spell checker is one example of such an integrated application package that operates under the NOS. A word processor might call on a spell checker to correct errors in a document. An e-mail application on the same workstation may also have its own spell checker; similarly a spreadsheet application may contain a third and separate spell checker. The current trend of "Office Suites" helps eliminate duplication by allowing several applications to share common tools, like spell checkers. Another source of duplication is applications from different vendors, because these applications will typically provide their own tools. This duplication could be avoided if developers agreed on standards, that would be supported by OSs. If only one tool is needed on a workstation, then applications should be able to access a single tool. Such standards are starting to emerge from vendors, but are not widespread yet.

As an example, Apple Computer has proposed a standard to permit free communication between small applications or "editors" while keeping the actual inner functionality of the applications proprietary. Apple Computer's latest developmental OS (MacOS 8) attempts this via a standard they call OpenDoc. "Editors" are placed in a central location that all applications can access and with a specific input/output format, but the heart of the application and the algorithm, remain proprietary.

Data duplication will not be significantly reduced until it is made easy for developers to do so. Methods to reduce data duplication are still not widespread and, in fact, market trends are actually working against the reduction of data duplication. A major factor contributing to the failure of software developers, as well as systems designers, is to embrace reducing data duplication is the increasing advancement in

hardware. As memory and hardware prices continue to decline, there is little need to go to the extra trouble of making programs share data. Increasingly there is more storage space for less money; vendors may not be concerned with integration applications. Cross-application interoperability as well as cross-system interoperability are essential to help reduce unnecessary data duplication. As discussed earlier in this chapter, there are methods available to achieve better data sharing. NOSs need to be designed to support these methods.

It is important to note that some duplication is not bad. In mission critical systems, even a small interruption of service is intolerable. Some built in duplication provides fault tolerance in the systems. One implementation of deliberate duplication that supports fault tolerance in a system is using a Redundant Array of Inexpensive Drives (RAID). RAID is a means of storing data in multiple locations so that if a hard drive "crashes", a backup is immediately available. Other methods of redundancy that can be purposely built into NOSs are backups at set intervals and disk mirroring.

b. Software maintenance costs

The next area of concern that TAFIM addresses regarding life cycle costs is software maintenance costs. Norman F. Schneidewind, a Fellow of the IEEE, defines maintenance as the "modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment." Furthermore, he states that there is a maintenance problem because:

- 75-80 percent of existing software was produced prior to significant use of structured programming.
- It is difficult to determine whether a change in code will affect something.

- It is difficult to relate specific programming actions to specific code. [SCHN87, p. 303]

He also makes the point that programmers cannot perform maintenance on systems which were not designed with maintenance in mind. Finally, he points out that good people have not traditionally been attracted to the field, stating that, "To work in maintenance has been akin to having bad breath." [SCHN87, p303]

Despite all the problems associated with software maintenance, it deserves our attention. A consultant from Anderson Consulting states that "for a system with a 5 year life, up front costs are [only]... 35-40%. If you had a 7-8 year life, it might go closer to [just]... 15%" [HANT96]. Up front costs include:

- Software license fees or salaries for programmers (if developed in-house)
- Hardware
- Network installation
- User training
- MIS training
- Implementation services (i.e. consulting) [HANT96]

This leaves the bulk of software costs come from the reoccurring costs from ownership. These include:

- Software maintenance fees
- Software upgrade licenses if the software was purchased (depending on the deal) or the cost of maintaining in-house and/or contracting programmers to maintain and enhance custom software

- Hardware upgrades (because "software always gets fatter and not skinnier.")[9]
- End user technical support
- Ongoing training for new users
- Operational costs such as hardware maintenance, making backups etc. [HANT96].

The idea that maintenance costs are the area in which the DOD spends the bulk of its information systems, is echoed throughout the book Software Maintenance Management, by Lientz and Swanson. Lientz and Swanson cite numerous examples of studies which place maintenance figures from 50 percent to 90 percent over the life cycle of a typical system [LIEN80, p. 4-5]. With most of the life cycle costs associated with systems attributed to reoccurring costs, it is obvious that DOD's efforts focused on maintenance would reap the most benefits.

In the past, the DOD developed much of its own software. Now, however, the DOD has realized that in most cases it is more cost effective to purchase software commercially. Commercial software developers, in most cases, have the resources to create applications that fulfill many needs of the military. In many cases, the DOD's needs are similar to commercial industry's needs. To that end, instead of custom developing applications, and maintaining DOD developed software, the DOD's current policy is to purchase COTS where appropriate. Instead of maintaining DOD-developed software, the DOD will purchase new or upgraded software (where possible), similar to businesses in the commercial sector. However, it is important to note that there is still a maintenance

---

[9] Mike Gancarz states in The UNIX Philosophy that one major mantra of the Unix programming community is that portable code always wins over compact code; after all, new hardware will be out next year that will run the software quicker.

function because frequent upgrades must be made and tailored to be compatible with the NOS (e.g., use of network drives, program and data sharing, use of drivers).

The U.S. government currently has several NOSs in use. Some, such as Unisys' CTOS NOS used by the Coast Guard, are used only by the Coast Guard and the developer. Consequently, the entire cost of upgrades, bug fixes, etc., are borne by the government. Some of these costs could be avoided, or reduced by migrating to commercial OSs.[10] Many of the life cycle costs associated with maintenance, although not eliminated, are reduced by such a migration.

The use of commercial NOSs would also mean a larger base of users, spreading the cost of the software product over the entire consumer base, including the U.S. Government. Distributing the costs over a large base of users, reduces software maintenance costs for the DOD over the life of the software. When purchasing commercial software products, the manufacturer (and subsequently all the users, not just the DOD) would bear the cost of developing, maintaining, and documenting the software product. Presumably there would be more product demand for software developed for the DOD and the commercial sector. This increased demand would provide more incentive for manufacturers to maintain and enhance software that contains the newest technologies.

Another concern is whether the DOD has the resources necessary to develop increasingly complex OSs, or even why DOD would want to. There are many OSs in the market place that fill most of the DOD's needs for automated information

---

[10] The Coast Guard is currently in the process of migrating from CTOS to Windows NT

systems. Vendors gain the advantage of economy of scale because of specialization and mass production.

Another way to reduce DOD software maintenance costs is to reduce the number of applications, OSs, and programming languages that are in use and perform similar or identical services. This area has historically been a source of high costs in the DOD. Figure 15 displays the rising dollar amounts of DOD software costs.

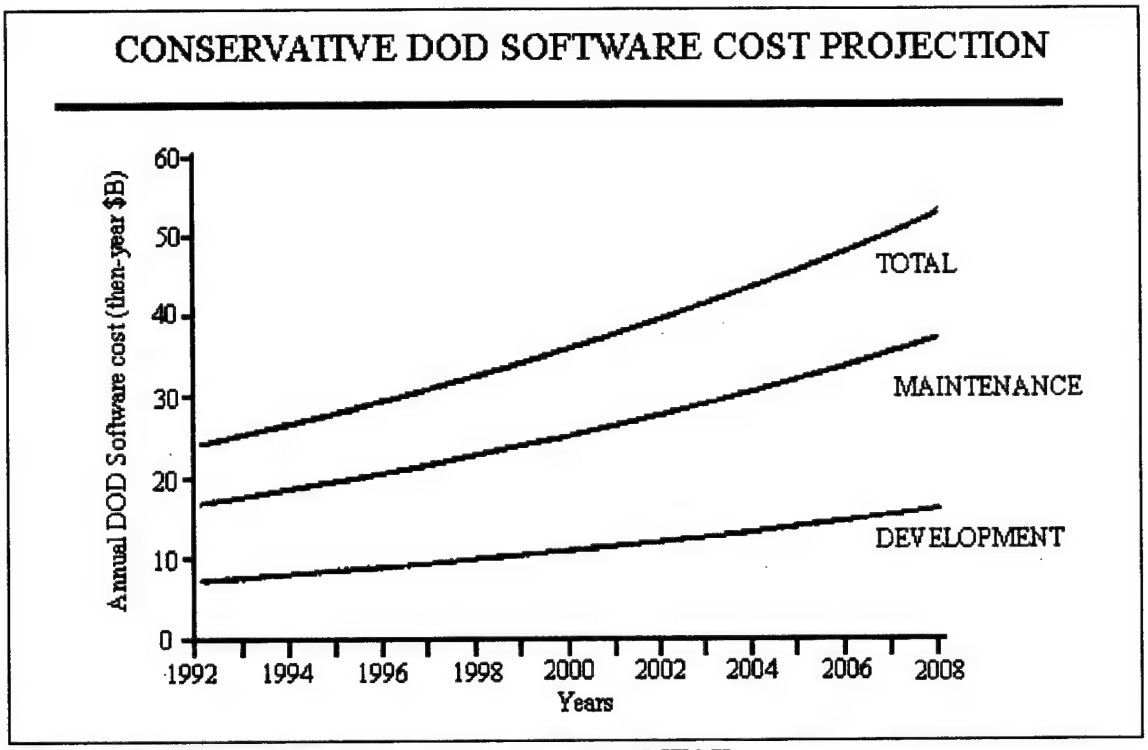

Figure 15. DOD Software Cost Projection [RAME95]

The more OSs, programming languages, and applications that the DOD supports, the higher are the costs associated with maintaining, developing, and training. This problem has been addressed in several DOD directives and programs; one example is the Ada mandate. The Ada mandate was an effort by the DOD to reduce the number of general programming languages in use, which had grown to 450 in the DOD. In fact, from 1968 until 1973 software costs increased in the DOD by 51 percent. The estimated

number of word processors in DOD use was said to be between 500 and 1500. Supporting so many applications, OSs and programming languages results in greater maintenance costs. If the number of NOSs and applications are decreased, then the number of trained personnel would decrease. [CS2970, p. 1-7]

c. Training costs

The last factor TAFIM addresses when describing methods to reduce life cycle costs is training costs. Anderson Consulting's list of life cycle costs was dominated by costs associated with training. Training costs are often difficult to measure. Not all training costs are spent as an outright expenditure in the DOD. It may be easy to attach a cost associated with sending personnel to a training program on a software product, but this leaves out many hidden costs.

End users gain the majority of their software knowledge through using a particular application. When users have to learn several different software applications, they learn best on the job. On the job training results in lost productivity instead of a outright expenditure and is one of the major hidden costs associated with training.

Training costs can be reduced by migrating to NOSs that are familiar to end users. Familiarity with a NOS may eliminate the necessity for users to participate in training programs. If users were initially familiar with a NOS, training could focus more on job related tasks, not basic familiarity with a NOS. Using more common industry wide NOSs would also increase the level of expertise in those NOSs within DOD organizations.

A familiar and intuitive GUI keeps training costs down, either outright expenditures or hidden costs. A familiar GUI should be one that a user encounters on the

job, as well as in the home and in formal schooling. Tasks that are difficult to perform due to non-intuitive interfaces will decrease the willingness of users to learn them.

The use of on-line documentation is another way to keep training costs down. Training manuals are helpful only if they are easily accessible. It is very time consuming for a user to search through indices. Space and weight are limited on ships and aircraft, so it is impossible to have a copy of every manual at every workstation. This makes having reference manuals on-line extremely desirable.

A benefit of on-line manuals is being able to quickly search for a given "keyword." Many software developers are going one step further and making items which actually walks the user through the steps to perform a task (Novell calls this feature a Coach, Apple calls it a Guide). Apple has taken this even further in their latest OS, MacOS 8.0, by simply asking for the information needed and letting the software actually perform the task. NOSs should be designed to not only promote easy task accomplishment, but also contain support methods that make it easy for users to inquire about how to do things.

Portable software also reduces the need for user training. Applications can be designed with the ability to be moved from one environment to another with little or no modification. This reduces the need for users to learn several versions of an application. A good example of this is WordPerfect. Corel ported WordPerfect from the Windows environment to the Unix environment. The program retains the same look and feel, as well as functionality. As APIs become more standardized, it is easier for applications to be ported. A NOS that can support such portability would reduce training costs.

3. Analysis of the Unix architecture

a. Data duplication

The Unix approach to avoiding data duplication and removing overlapping functionality can be best described by understanding how Unix was developed. Unix was developed as a multitasking, multithreaded NOS. From its origin, it was developed as a multi-user NOS [GASK95, p. 1086]. Because of the need for flexibility to support multiple users, Unix needed an easy way to share data. Methods within Unix that support data sharing were described earlier under user productivity. The sharing of data helps to remove unnecessary duplication.

Interoperability provides another method for reducing overlapping functionality and data duplication. By allowing all computers on a Unix network to communicate via the same set of standard protocols, like TCP/IP, information can be shared throughout the network. Information that can be obtained by accessing remote computers throughout the network will no longer be required to be stored on local workstations. Physical network connections are not enough to accomplish this. Interoperability and connectivity are the keys in reducing the need for organizations and commands throughout the DOD to store the same information.

b. Software maintenance

In a Unix system there are several built-in methods that make software maintenance relatively easy. The history of Unix accepting and promoting *open* systems standards and its standing within the academic community also helps limit maintenance costs.

The Unix architecture provides life cycle cost savings, such as the use of pipes. Pipes are a Unix characteristic that help promote a reduction of software life cycle costs. Pipes are a means of porting the output of one file to the input of another file without the use of a temporary file. In Unix, everything is treated like a file, and all files are treated as a stream of ASCII characters. The keyboard in known as the *stdin* (standard in) file and the monitor is the *stdout* (standard out) file. Since everything, including devices, is treated as a stream of characters, it is easy to view and manipulate data.

Programs are considered merely as filters of data, not creators of data. In order for these filters to be easily created, understood, and modified, they must be kept simple and small. If a large task is to be kept simple, it must be broken into many parts. If a program is comprised of many parts, it must be easy to pass data back and forth between these parts. Pipes make passing data between modules simple. These small modules make modifications easier, because the programmer must only understand one module at a time as modifications are made.

The loss of funds spent on software that is never delivered is another problem that haunts the DOD. Because applications in the Unix environment are merely a collection of smaller modules and the pipes that connect them together, programmers have the ability to isolate and trace data as it flows through the program, module to module, during the early stages of prototyping. If problems are encountered, isolation of modules is possible by looking at the data, via pipes, before and after each module to determine problem locations. This ability to isolate problems and rapid application prototyping

speeds up the development process as well as ensuring that the final product is exactly what the user wants. [GANC95, p. 58]

Software maintenance has always been a complex and cumbersome task, which becomes more complex as the number of copies of an application increase. Due to the multi-user, host-based environment in which Unix operates, Unix programs are designed to be network centric. This allows network administrators to install a single copy of a program on a server and have clients access this copy as necessary. Preferences (such as the appearances of the toolbar) are stored in the home directory of the current client (user, not machine). This single copy (sold with a site license for the required number of users) is relatively easy to upgrade and maintain, thus reducing time and labor costs.

With *open* systems and a set of standards (e.g., POSIX), the industry follows a set of standard practices that allows IT managers to develop software modules that have "black box functionality". Functionality within the box becomes secondary to ensuring the box delivers the desired outputs. This principle applies to NOSs. Treating NOS components as black boxes reduce maintenance costs across multiple software and hardware vendors.[11]

As stated earlier, the POSIX standard allows programs to use a standard set of published APIs to provide a standard interface with the NOS. A standard set of APIs will greatly reduce the complexity of developing software on different NOSs, which

---

[11] The authors note that this level abstraction is extremely difficult to achieve and not present in the market today.

currently have their own proprietary set of standards. A common set of APIs means reduced complexity, increased ease of maintenance, which reduces life cycle cost.

Since the original purpose of POSIX was to define a standard *open* interface, based on the Unix system, Unix systems comply with, or are easily adapted to meet this portion of the open system requirements for the DII COE. In fact, the whole intent of the original Unix design was to be open, but it has been changing for years and branching in several slightly different directions. Because of this divergence, Unix vendors have been eager to adopt open systems standards to allow applications to function on different variants of Unix and to converge on a set of open system standards. Spec 1170 is one example of a recent open Unix based standard that attempts to achieve this objective. It is intended to provide standards that will allow organizations to mix and match Unix NOSs and platforms compatibly. [WEBS94]

Basing an information system in the DOD on Unix enables the DOD to capitalize on academic resources. Since Unix source code has historically been readily available and inexpensive, it has flourished in the academic environment. "[Unix] is the undisputed system of choice in the academic world." [GANC95, p. xix] Because Unix was written in a programming language eventually called C, it has traditionally come packaged with its own C compiler. Even today, Unix NOSs come with a C compiler; for example Sun Solaris v2.3 comes packaged with a Sun C compiler. Not surprisingly, programming courses in C and C++ are the most common programming languages colleges and universities teach in the US. However, it should be noted that applications written for Unix are several times more expensive than their Windows counterparts.

The preponderance of Unix in the academic world has produced a wealth of Unix experience. Programmers, majoring in computer science, right out of college have experience writing code for the Unix environment. Although it is true that C and C++ are portable, changes must be made to code when switching from the NOS that the code was written for to another NOS. The more specialized the NOSs the DOD has, the more specialized the training must be, and the greater the cost to the DOD.

c. Training

Unix source code was made available to other groups within AT&T and, for educational purposes, to universities [MICR03, p. 2]. The academic community has produced Unix system administrators for years. Unfortunately, government salaries are such that it is difficult to attract people with Unix skills. If it were possible to hire these people in large numbers, training costs would be reduced. The reality is that the government has to provide training in Unix, either in-house or under contract.

Another aid in understanding Unix is the vast amount of information published on the Internet. Since Unix has been available for many years, coupled with its *openness* and academic ties, much has been published to aid others in understanding the system. These documents can be useful tools in simplifying Unix administration.

4. Analysis of the Windows NT architecture

a. Data duplication

Microsoft takes an approach similar to Unix in avoiding data duplication. Windows NT is not a true object-oriented NOS, but it does represent internal system

resources as objects. This helps reduce data duplication as described below. Microsoft defines objects as a combination of three traits:

- Attributes in the form of program variables that collectively define the object's state.
- Behavior in the form of code modules or methods that can modify those attributes.
- An identity that distinguishes one object from all others.

Objects communicate by a form of message passing. This message passing system is analogous to the Unix file metaphor and pipes, only more powerful. NT treats all things that Unix calls files as objects, but also includes "processes and threads, shared memory segments, and access rights." [NTUX95, p. 5] OLE is a method provided and supported by Windows NT to share these objects between applications. By treating everything as objects, a program can pass information between processes, thus reducing redundancy. These feature help provide data sharing between applications, eliminating the need for data duplication.

b. Software maintenance

Traditional client/server PC networks require the installation of executable programs at the server and support files at the client. NT does not change this paradigm; software must still be installed at both locations. It is also important to note that software can be designed for the client/server system in different ways. One method is to have a portion of the program reside on the server and called upon as necessary. A client acts like a dumb terminal. Limited pre-processing sometimes takes place on the client, but the heart of the processing takes place on the server. Also, Windows NT allows the installation of client portions or even entire stand alone applications to be installed from

the server. The application can be "pushed" or "pulled" along the network to the client machine where it will reside on the hard drive. This prevents the system administrator from having to be physically present at all machines in order to install programs. This saves time and money, and this savings grows as upgrades, new programs, and additional workstations are installed.

Power failures and brownouts are a fairly common occurrence. Universal Power Supplies (UPS) provide emergency power in such an event. Many UPS devices today allow for direct connection to the NOS. This allows the NOS to give connected users notice of the power failure to allow an orderly shutdown and save data. UPSs also notify the NOS that power has been restored and that shutdown is not necessary. Windows NT provides a serial port connection to connect to these types of UPS devices. Windows NT can also be configured by the network administrator to prevent any new connections during the time period the UPS is providing power. This makes the maintenance job of the administrator must easier, because time is not spent trying to recover lost data.

c. Training

Microsoft has been able to increase its market share of OSs in an expanding PC market. More people are becoming computer literate, and the vast majority of the PCs in the market place are WINTEL (Microsoft Windows running on Intel processors). This dominance is mirrored in the DOD. A recent Government Computer News Survey (GCN) of over four thousand DOD personnel who identified themselves as purchasers and users of OSs for desktop computers indicated that they rely on Microsoft OSs at a rate of 28-1

over the next popular operating system (Interestingly enough, MacOS was second and Unix did not even garnish enough responses to be rated) [GCNP96, p. 20]. While there is greater use of Windows for Workgroups and Windows 95 than Windows NT, the survey shows the base of Microsoft users in the DOD. This has several implications for training personnel.

With such a large base of Microsoft OS users in the DOD, there is a substantial amount of knowledge and experience with Windows products. The GCN survey identified Windows 95 as second best in ease of use, following MacOS. For users who are experienced with Windows 95, this could mean a substantial savings in training when moving to Windows NT, because Windows NT 4.0 has the same interface as Windows 95 and Windows NT 3.51 can be updated to the Windows 95 GUI with a free download. Taking advantage of this installed base would alleviate some of the difficulty associated with transitioning to a different NOS.

Not only would some users be familiar with the interface of the Windows NT, they would also have the benefit of having worked with Windows applications. For common applications, like word processors and spreadsheets, DOD would reduce training costs. Granted, there will be some mission specific applications that will be new to users, but they will at least be familiar with the Windows based interface.

The powerful help features in Windows NT reduces training time and costs. The help features are easy to use in Windows NT for even the inexperienced user. Help is available in the form of indexes, search tools, and hyper-text. Hypertext help is available so that the user can access information from the desktop, rather than searching through manuals. With application developers following these help standards, users will

be able to find answers to their questions faster, be more productive, and learn while on the job.

Microsoft also provides another method to train users at their workstation via Wizards. Wizards are used to provide the user with the basic steps needed to accomplish a task. They are used to help the user to accomplish a variety of tasks, including installing software, changing GUI features, and performing tasks within applications. Wizards ask the user questions on what task he wants to perform, and the information that is needed to perform the task. Some of the more powerful wizards even perform the task for the user rather than walking the user through the task. This style of user help greatly reduces the burden on the user. If the user is new, the wizard can walk him through the task so he gets hands-on experience of how to do the task, using sight, sound, and even video. The user in effect, learns while doing.

5. Summary of findings

Windows NTs use of the object metaphor allows increased data sharing when compared to the Unix environment. While Unix does provide for data sharing via pipes, the Windows NT implementation of this is much more robust and powerful. This increases Windows ability to increase data sharing and reduces data duplication, thus reducing life cycle costs.

Perhaps training is the most significant factor in life cycle costs. Given that the DOD is such a large organization, it is important to implement NOSs and applications that users will be able to adapt to and be productive with. Windows software is the best selling OS in the world today, with Windows 95 and Windows NT leading the way. The home computer revolution has led to Microsoft's selling a projected 70 million units of Windows

OSs in 1996 [COMP96]. Windows NT is even predicted to outsell all Unix variants combined for server application by the end of 1996 [EETI96]. Figure 16 shows the dominance of Microsoft OSs in the market today, and the predicted market dominance of Windows OSs, particularly the increasing market strength of Windows NT. This market presence could be a significant advantage for the DOD, potentially serving to reduce the cost of training.

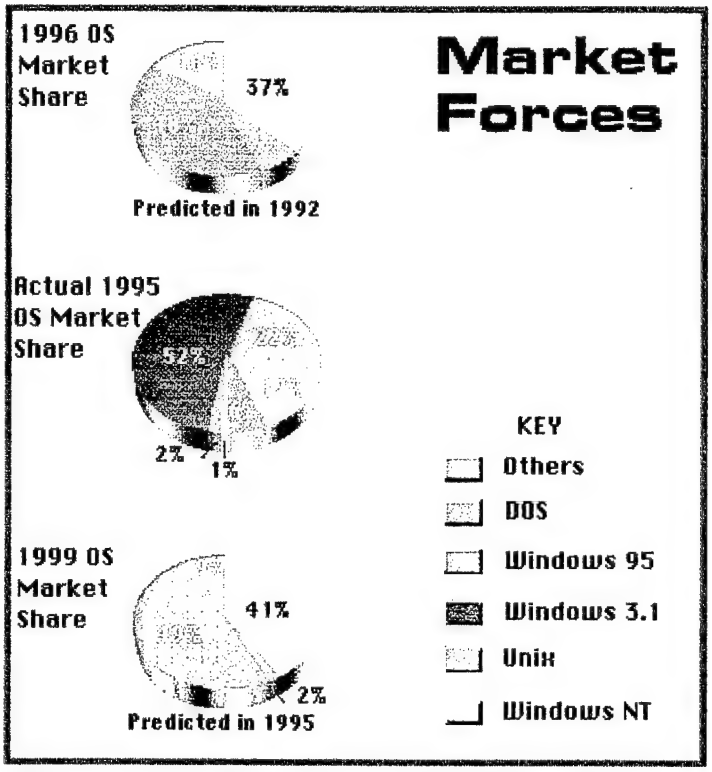

Figure 16. OS Market Share [HALF96, p. 52]

While Unix NOSs are dominant in universities, PC systems are prevalent at the primary and secondary schools. Even with the dominance of Unix in universities, there is also a trend to require each student to own a PC. This increases the likelihood that many members of DOD will have been exposed to Windows. A minority of personnel in the

service have been exposed to Unix, and fewer still are literate in Unix. With such a dominate place in the market, exposure to Window's GUI is virtually guaranteed.

Unix was first developed with the idea that its users would be computer literate. In fact, the "designers of Unix took an inhospitable 'if you can't understand it, you don't belong here' kind of approach." [GANC95, p. xvii]. This mentality, on the so called intuitiveness of Unix, underscores one of its inherent weaknesses. In an era where the trend in computing is to make computers more useful to users, this mentality will not survive in the market place.

When considering reducing life cycle costs, it would be foolish for the DOD to ignore the savings in training costs that would be achieved by using a Windows NOS. On the other hand, if Unix were the dominant NOS in DOD, it would require a substantial training effort, resulting in substantial costs and reduced productivity.

In conclusion, Windows NT is the more cost effective platform when considering the life cycle costs of a system based on the DII COE. This conclusion is reached using the TRM objective as the means of comparison. This is due to reduced requirements for training and data duplication.

G. IMPROVE SECURITY

1. TAFIM's definition of the objective

TAFIM states that security in information systems that may need to operate simultaneously in various DOD environments (tactical, strategic, and sustaining base) will be improved in DOD information systems by satisfying the following principles:

- **Uniform Security Accreditation and Certification.** Uniform certification and accreditation procedures will not only reduce the time needed to approve system operation but will result in more consistent use of security mechanisms to protect sensitive data.

- **Consistent Security Interfaces.** Consistent security interfaces and labeling procedures will reduce errors when managing sensitive data and reduce learning time when changing from system to system. Not all mission-area applications will need the same suite of security features, but any features used will be consistent across applications. Users will see the same security labels in a common format and manage them in the same way.

- **Support for Simultaneous Processing in Single Platforms of Different Information Domains.** Security protection will be provided for simultaneous processing of various categories of information within a single system. Information systems that can support multiple security policies can support multiple missions with varying sensitivity and rules for protected use. This will include support of simultaneous processing under multiple security policies of any complexity or type, including policies for sensitive unclassified information and multiple categories of classified information. This type of support will also permit users with different security attributes to simultaneously use the system. Separate or dedicated information systems for processing information controlled by different security policies will be reduced or eliminated.

- **Support for Simultaneous Processing in a Distributed System of Different Information Domains.** Security protection will be provided for simultaneous processing of various categories of information in a distributed environment. This protection will apply to processing of information controlled by multiple security policies in distributed networks using heterogeneous platforms and communications networks. This will greatly extend the flexibility of the system implementor in providing cost-effective information systems based on open systems principles.

- **Support for Use of Common User Communications Systems.** Security protection will be provided in such a way as to permit use of common carrier (public) systems for communications connectivity. It will also permit the use of Department-owned common user communications systems. This use of public and Department common user global communications networks will result in the potential for enhanced cost effective interoperability across mission areas. [DISA03, Vol. 2 p. 2-4]

2. Interpretation of the objective

a. Uniform security accreditation and certification

The federal government has over the years developed a uniform computer security accreditation and certification system. In August of 1983, the federal government released the DOD Trusted Computer System Evaluation Criteria (TCSEC). The TCSEC is more commonly referred to as the Orange Book. The Orange Book was a response by the federal government to growing concerns about computer security. It was felt that a standard was needed for the purchase and use of computers in the federal government. This would develop consistency in security features across government systems.

The Orange Book defines four hierarchical divisions of security protection. The divisions are called: minimal security, discretionary protection, mandatory protection, and verified protection. The four divisions correspond to the letters D, C, B, and A respectively. Each division consists on one or more classes, corresponding to a greater degree of security. These divisions are designated by numbers. Some classes have only one division, others have up to three. The higher the letter and number, the more secure a system is considered. Each class is defined by a specific set of criteria that a system must meet to be awarded a rating in that category. The criteria fall into four categories: security policy, accountability, assurance, and documentation. Appendix C contains a table which details the requirements at each level of security.

The Orange Book states that its evaluation criteria were developed based on three objectives: measurement, guidance, and acquisition. It was developed to provide users with a metric with which they can access the degree of security in a system. A user

can be assured that a system that has a B2 security rating is more secure than a system that has a C2 rating. The Orange Book also provides developers guidance to build systems that satisfy government security requirements. The Orange Book also provides a clear way to specify security requirements for systems, making it easier for government agencies to specify requirements in acquiring systems. [RUSS92, p. 104-105]

The Computer Security Act of 1987, Public Law 100-235, was designed to improve the security and privacy of federal computer systems. It specifically stated requirements for the minimum security requirements for federal computer systems. It mandated that all federal computers meet the minimum requirements of a C2 classification, as outlined in the Orange Book. The law requires that all computer systems that handle "classified and/or sensitive unclassified information ... shall implement required C2 security features by 1992." [CONS01, p. 8]

Navy Standard Operation Procedure document 5239.15 (NAVSOP-5239.15) contains the functional requirements for C2 class systems. This instruction is called the Controlled Access Protection (CAP) guidebook. This guidebook describes the minimum set of automated controls for DON information systems. Since all DON systems are considered to process sensitive unclassified data as a minimum, they must adhere to C2 security requirements as outlined in CAP.

In order to be certified C2, a computer system must meet the requirements described in the Orange Book for a C2 system, in addition to fulfilling the requirements for a D and C1 system. D certified systems are systems that are minimally secure, the Orange Book lists no requirements for this class. C1 systems have limited security features. The security features of this class are mainly intended to keep users from making security

violations. C2 systems provide more stringent security features than C1 systems. C2 systems are systems considered to provide controlled access protection. [RUSS92, p. 156] A C2 secure system offers increased security features in the following areas:

- Discretionary Access Control
- Object Reuse
- Identification and Authentication
- Audit
- System Architecture
- Security Testing
- Documentation

In order to be considered C2, a system must meet the criteria stated in the Orange Book for each of the areas.

Discretionary Access Control (DAC) is a policy which restricts access to files based on the identification of users, or a group, to which they belong. This method is in contrast to Mandatory Access Control (MAC), in which the system controls access; DAC is applied at the users' discretion. Essentially, a system that uses DAC and is C2 certified must have the ability to distinguish between users.

Object Reuse requires that a system be able to "protect files, memory and other objects in a trusted system from being accidentally accessed by users who are not authorized to access them." [RUSS92, p. 118] Where as DAC assigns who can and cannot access an object initially, Object Reuse controls these features when objects are reassigned. This is an important feature for a NOS. Object reuse features may include maintaining a file containing the identifications of users deleted from the system. When a

new user is added, Object Reuse ensures that the new identification of the user does not duplicate the access rights of the same previously deleted user. [RUSS92, p. 118] It also prevents users from accessing memory which is being used by another user.

Identification and Authorization is a part of all security levels, but it is increased at the C2 level. Basically, this requirement mandates that every user has a unique account name and password. Both of these must be supplied before gaining access to the system. [RUSS92, p. 124-125] The Orange Book does not state how passwords should be protected. Passwords remain essential for secure systems.

All passwords are not created equal. Two early computer pioneers, Robert Morris and Ken Thompson, studied passwords and found that 86 percent of the time they could guess correctly by using family names, birthdays, street names, common English words, etc. [TANE92, p. 189]. Their research demonstrates how easily passwords can be determined if not chosen carefully. Equation 1 shows the importance of requiring passwords of suitable length. It determines what the length a password should be, provided that the probability of guessing it is one in a million. The probability P of guessing a password is given as:

$$P = \frac{G}{S} = \frac{L * R}{S}$$

Equation 1. The probability of guessing a password

where L is the lifetime of the password, R is the number of guesses per unit time that can be made, S is the total number of unique passwords that can be generated, and G is the total number of guesses that can be made in the password's lifetime. Equation 2 solves for S, using a lifetime of six months, one guess every second, and a one in a million chance at getting it right yields:

$$P = \frac{G}{S} = \frac{L*R}{S} \Rightarrow S = \frac{L*R}{P} = \frac{183(\text{days/6months})*86400(\text{guesses/day})}{0.000001} = 1.58112*10^{13}$$

Equation 2. Determining the number of possible passwords

S then is the total number of possible passwords. From S, we can determine the password length necessary, stated in Equation 3 as M, to satisfy our given constraints. Now given that A, the set of possible characters which M is comprised of, contains 94 possible characters (standard ASCII keyboard characters), then

$$S = A^M \Rightarrow M = \frac{\log S}{\log A} = \frac{\log 1.58112*10^{13}}{\log 94} = 6.69 \; characters$$

Equation 3. Determining the length of a password

From the parameters that have been chosen, passwords need to be 7 characters long to be considered good, and these characters are drawn from lower and upper case letters, numbers, and other keyboard characters.

Audit trails are also necessary for a C2 level secure system. In computer systems, auditing is the ability to record, review, and examine all security-related activities in a trusted system. The primary reason for audit trails is that even the most secure systems are vulnerable to attacks and audit trails are an excellent way to determine whether an attack has occurred and how the attack was attempted. [RUSS92, p. 128-129]

System architecture falls into what the Orange Book calls the assurance category of a secure system. Although a C2 system's architecture does not need to be designed specifically for security, it must be designed using sound principles. These include basic concepts like protection of resources and separation of user and system functions. [RUSS92, p. 134] It also includes features to keep users out of memory areas where they do not belong.

System Integrity describes the concept that hardware, firmware, and software must work, and be tested to ensure that they will continue to work, properly. One aspect of System Integrity involves correct initialization of system resources. This requirement is usually satisfied by vendors providing system tests during startup that are included in CMOS.

Security Testing involves evaluating a system to determine whether the system functions as described in the documentation. There are two basic types of security testing: mechanism and interface testing. Mechanism testing is the testing of the security mechanisms provided by MAC, auditing, labeling, and authentication and identification. Interface testing involves testing all the user actions which request security functions. [RUSS92, p. 142]

The final category described in the Orange Book is documentation. This is sub-divided into various sections, but the basic requirements for a C2 system are that a manual be provided which explains why security is important, how to administer DAC, how to administer identification and authentication, and how to administer auditing capabilities [RUSS92, p. 151]. The security documentation provides a description of the manufacturers' view of security for the system administrator.

Fundamentally, a C2 certified information system must protect systems resources via access control features. TCSEC provides a standardized method for evaluating systems against a defined set of requirements. This satisfies TAFIM's intention of having a uniform security accreditation and certification process. However, the requirements set forth in the Orange Book for a C2 certified system only meet part of the requirements that TAFIM describes later in this objective. "C2 does not have any

provision for viruses, control encryption, integrity checking, network interconnections, or remote accessibility. Any computer professional working for a major organization knows that without these features, security remains virtually non-existent." [SCHW95] Similarly, the security demands of the DII COE are far greater than security measures contained in C2 level systems. These additional needs are outlined in the remaining parts of the objective.

b. Consistent security interfaces

Consistent user interfaces not only results in lower training costs, but enhance understanding of security procedures. The demands of the DII COE require that in many cases the network be easily and quickly managed by non-IT professionals. This requires easily understood and administered security features. Again, a GUI should be consistent and intuitive throughout, and the interface to the security features should be consistent with the GUI

c. Simultaneous processing at multiple security levels on a single platform

A system which handles multiple information classifications at a number of different security levels within a single security system is called a Multiple Level Security (MLS) system. Two things are required for a MLS system: MAC and sensitivity labels. MAC is an access policy which assigns sensitivity to all subjects and all objects within a system. Sensitivity labels define the required level of trust that a user must have in order to gain access to a file or object.

In contrast, most *secure* systems today operate in what is known as system high. In these types of systems, all users are assumed to have the same security clearances. This clearance is equivalent to the highest level of security of the information being processed on the system. MAC and sensitivity labels are only required for B2 certified systems and higher. This shows that while the common perception is that government computer systems are required to be C2 certified, TAFIM requires other security features which are typically associated with B-level systems.

d. Simultaneous processing in multiple security levels on a distributed system

This tenet contains essentially the same requirements as above. The primary difference is that this tenet extends the MLS ideas expressed above to a distributed system. An additional caveat is that different platforms must be able to process the data within the same system. This is primarily a function of interoperability, but raises new security issues. One potential security concern is whether the NOS can determine if the computer logging on to a network is really the computer it says it is.

e. Use of common communications systems

The idea behind this tenet is that the DOD can use Plain Old Telephone System (POTS) and other public means, rather than DOD or other government lines. Government owned lines can be monitored more easily to determine tampering, eaves dropping, or other security breaches, but this is more difficult on POTS. Communicating securely over public lines requires some form of encryption, in which both the sender and the receiver have a key to the encryption method. Not only can encryption provide a

measure of security, it also provides limited authentication verification capabilities as well as data integrity. There are very few NOSs which provide encryption as a standard feature. Those that do are generally specifically developed for security purposes. HP-UX and Windows NT both lack point-to-point encryption and therefore both fail to meet the criteria outlined in this tenet.

3. Analysis of the Unix architecture

Originally, Unix had no security features. In fact, when Unix was developed, it was the general feeling that security was an impediment and counter-productive to its purpose. Unix was designed to allow users and programmers to work in an interactive manner; security features merely slowed down this interaction. As Unix proliferated into the academic and scientific communities, the need for security was apparent and features were added.

a. Uniform security accreditation and certification

It is important to note that a C2 certification must include the computer as well as the NOS. With this in mind, HP-UX was certified as a C2 level system on a TAC-4 HP workstation [HPVI96]. In addition to the features used to satisfy the C2 requirements, HP-UX has many additional features.

The first additional security feature is improved auditing. HP-UX can detect actions that try to introduce or delete objects into a user's space. The audit record will automatically include the name of the object and the ID of the user who performed the insertion or deletion. This feature helps the system administrator track down illegal or

unwanted file deletions as well as attempts to post documents to an account by someone other than the owner.

Access Control Lists are a portion of the DAC feature of security that HP-UX has added as a standard feature. Access control lists are lists of each named object and each user with access to the object. HP-UX's Access Control Lists are also capable of specifying a list of named individuals and a list of groups of named individuals for which no access is given.

Additional capabilities for password management are also included in HP-UX. These include the ability to maintain an encrypted password database which only the superuser (the system administrator) has access to. Additionally, the NOS can generate *good* passwords, screen user generated passwords for those which are easily cracked, and enforce the concept of password aging to force users to periodically create new passwords.

Another feature that HP-UX supports for increased security is the ability to control the log in times and dates that users are allowed to access the system. HP-UX can also restrict where a user can and cannot log in from. This prevents a user from logging in and using resources at times when the system administrator determines is not conducive to the system.

The final feature that the HP-UX Unix variant supports is called boot authentication. This restricts the ability of anyone without clearance from booting up the system. The intention of this feature is to prevent anyone other than the system administrator (or a designated alternate) from booting the system. It also prevents

someone from booting from a floppy and copying information without invoking the NOSs other security features.

b. Consistent security interfaces

A consistent security interface for HP-UX is provided by System Administration Manager (SAM). This program is discussed in detail in the next section and will not be discussed here. SAM provides a consistent interface for the system administrator to interact with security features. Additionally, the SAM interface can be used by third party programmers in their products to provide further consistency.

c. Simultaneous processing at multiple security levels on a single platform and distributed system

MLS is not supported by HP-UX. Although HP-UX implements B3 level DAC, it does not implement Labels or MAC and therefore cannot be used on a MLS system, either stand alone or in a distributed system. It must be used on a system-high type of computer network with restricted physical access, which is isolated from networks of different levels of security.

d. Use of common communications systems

HP-UX fails to provide any method of secure communications when using public communication systems. As discussed previously, this would entail using some type of encryption between computers. Information that is being transmitted over telephone lines, either by e-mail or file transfer, is vulnerable to interception. At present HP-UX provides no capabilities to make this type of transmission secure. This potential threat must be addressed by third party products.

4. Analysis of the Windows NT architecture

Unlike Unix, Microsoft boasts that Windows NT was built from the ground up with security in mind and developed to meet security requirements of the U.S. government. [MICR13] Although this is in sharp contrast to the origins of Unix, Windows NT provides about the same degree of security protection.

a. Uniform security accreditation and certification

Windows NT was certified at the C2 level on July 31 1995, but there are certain conditions to this certification. As previously mentioned, a C2 classification must be performed on a computer system, an OS/NOS alone cannot be certified. Windows NT was certified on a machine with a disabled floppy drive. Booting from a floppy drive enables access to hard drives without invoking Windows NT's security features [CONS02, p. 8]. Microsoft recommends the following actions to allow NT to retain its C2 rating:

- Keep all files on file servers. Also, keep the file servers themselves under lock and key in a "glass cage" or closet, with access provided only to administrators.
- Disable all floppy drives on the server. [MICR10]

To qualify for a C2 certification, the NTFS (NT File System) must be used instead of the File Allocation Table (FAT) format, which allows DOS compatibility. Windows NT can read and write to FAT partitioned (i.e., MS-DOS formatted), but these partitions do not meet C2 requirements. This is due to the fact that FAT partitions can be shared or not shared, but do not allow restricting local file access like NTFS partitions do. [RULE95, p. 69]

Some of Windows NT's additional security features which are not covered by the C2 certification were examined against the B2 Trusted Path and B2 Trusted Facility Management functional requirements of the TCSEC. In order to satisfy the B2 Trusted Path functional requirement, a system must support a trusted communication path between itself and the user for identification and authentication. Although Microsoft states that Windows NT satisfies these functional requirements at the B2 level, it was not evaluated against any assurance requirements above its rated C2 level security by TCSEC. [MICR12]

Windows NT provides audit trails, including one for printers, allowing the system administrator to make printers more secure. This audit trail is similar to those for access to the system, in that it provides recording the use of the printer, time of use, and the account that used the printer. This helps the system administrator track down improper use of printers [RULE95, p. 207]. These audit trials can also track port usage as well as which printer printed a certain document. [SCHW95]

Windows NT is also capable of using an Access Control List. This is a feature not required by C2, but is nevertheless an important security requirement. Access Control Lists are not required until the B3 level is reached. Although Windows NT uses ACLs for increased security, it does not incorporate the entire requirements for ACLs at the B3 level. The B3 requirements for ACLs include the ability to specify a list of individuals with access and the level of access for every named object in the system [RUSS92, p. 290]. Windows NT does not provide the granularity to list each user who has access rights to a specific object; although a global access list can be obtained.

b. Consistent security interfaces

The security interface to Windows NT is the familiar Windows interface. With Windows NT, user accounts are managed centrally. The administrator can specify group memberships, logon hours, account expiration dates, and other user account parameters via graphical tools. The administrator can also audit security related events such as user access to files, directories, printers, and other resources and logon attempts. The system can even be set to "lock out" a user after a predetermined number of failed logon attempts. Administrators can also force password expiration and set password complexity rules so that users are forced to choose *good* passwords.

A simple password-based logon procedure gives users access to the appropriate network resources. Windows NT uses a system-level encryption of the user's password, so that it is never passed unencrypted over the wire. This encryption prevents discovery of a user's password through wire "sniffing." [MICR13]

c. Simultaneous processing at multiple security levels on a single platform and distributed system

Similar to HP-UX, Windows NT does not support MLS. Without the features of labeling and MAC, a computer system cannot be used securely at different security classification levels. This is true regardless of whether or not the process takes place locally or in a distributed environment.

d. Use of common communications systems

Similar to HP-UX, Windows NT does not provide any features that provide security when transmitting information across public communication lines. The

only Windows NT feature that is useful for increasing Identification and Authentication security features in communications over POTS is the Remote Access Service (RAS). RAS supports a feature called callback. Callback allows the server to call users back at a predetermined number to verify connection to the network, after the user calls the server but prior to being logged on by Windows NT. This feature prevents remote access of a computer from any location other than those approved by the system administrator. This adds another small measure of security. [RULE95, p. 379]

5. Summary of findings

TAFIM requires that information systems have stringent security features, in fact the TAFIM requirements exceed recent federal government requirements. Currently, government systems require that DOD information systems be C2 certified. This federal requirement helps satisfy TAFIM's objective to have a uniform certification and accreditation procedure for information systems. The Orange Book provides for this and the Red Book defines these requirements in a network environment. The problem lies in the fact that other security requirements described in TAFIM are characteristic of systems with security classifications greater than C2. This makes the government mandated requirements for C2 certification of DOD information systems inconsistent with TAFIM's security objective.

The feature TAFIM specifies which would improve NOS security are those that support a MLS. MLS features are those that are characteristic of systems that the Orange Book categorizes as B-level certified. These requirements are more characteristic of the security needs of DII COE implementations for tactical support functions. The increased

security requirements of TAFIM over those covered by The Computer Security Act of 1987 make it more difficult for commercial developers to satisfy TAFIM requirements.

Both NOSs, HP-UX and Windows NT, have been certified to meet the requirements outlined in the Orange Book for C2 certification. However, an important distinction is that these systems are NOSs and they need to be evaluated against the Red Book, which contains an interpretation of the Orange Book requirements for networks. Windows NT is currently undergoing this evaluation, but the process is both long and expensive.

Although both systems meet the requirements for C2 certification, it appears that Windows NT was designed simply to meet the certification process and leaves the rest of the requirements needed for a secure OS/NOS to third party vendors. This highlights one difficulty of basing a DOD information system on a commercial NOS. The customer base for a NOS in this case is the entire market, not the DOD alone. Consequently, specific DOD security requirements that may be peculiar, or unnecessary for the rest of the market, are often left out of the product. The caveats for keeping Windows NT secure, removing floppy drives and using only NTFS, eliminate two of Windows NT's many strengths: DOS legacy compatibility and a PC friendly environment.

Both HP-UX and Windows NT offer additional security features, like an ACL, but HP-UXs features are more robust and offer better value. The Access Control List provided in Windows NT cannot "map users ... against logon permission times or dates." [SCHW95]. Both HP-UX and Windows NT extra utilities make a more secure product, but neither vendor wanted to alienate commercial customers with security that was either unnecessary for corporate customers or too difficult to maintain.

While both provide additional security features that go beyond C2 level requirements, neither NOS comes close to supporting the types of features required for MLS on either a single platform or in a distributed environment. Windows NT was "designed with security in mind," and in fact the National Security Administration (NSA) is paying to have a feasibility study on ways to improve its security. [MCCA96, p. 6] Variants of Unix have been tested at the B2 and even B3 level, but not HP-UX.

The addition of the callback feature to allow pre-approved numbers to be called back remotely, is a nice to have feature, but again, falls far short of the desired capabilities required by the TAFIM tenet. Neither NOS meets the need for communication over POTS without help from third party software. The government currently restricts sales of encryption software outside the U.S., so developers are reluctant to incorporate these features into products which they hope to sell world-wide, like a NOS. Also, the additional time, money, and effort to test and certify these products at the B-level of security is a big deterrent to developers, particularly when they feel that the commercial sector does not want or need some of those feature, when the U.S. Government only requires C2.

Ideally, the NOS selected for information systems based on the DII COE would provide the security features described in TAFIM without the need for an additional third party solution. Why would the DOD want to buy a NOS that needs additional software to make it TAFIM compliant? Consequently, the DOD needs to acquire a NOS with the security features that meet its needs. While both Windows NT and HP-UX fail to meet several of the security requirements described in TAFIM, the HP-UX security approach does a better job of meeting government requirements for C2 certification. This will

dramatically shift in favor of Windows NT if it obtains a B-level certification which does not require modifications to the extent that it would no longer be compatible with its base of existing software.

H. IMPROVE MANAGEABILITY

1. TAFIM's definition of the objective

TAFIM's states that improving manageability can be realized by applying the following principles:

- **Consistent Management Interface.** Consistency of management practices and procedures will facilitate management across all applications and their underlying support structures. Users will accomplish work more efficiently by having the management burden simplified through such an interface.

- **Management Standardization.** By standardizing management practices, control of individual and consolidated processes will be improved in all interoperable scenarios.

- **Reduced Operations, Administration, and Maintenance (OA&M) Costs.** OA&M costs will be reduced through the availability of improved management products and increased standardization of objects being managed. [DISA03, Vol. 2 p. 2-5]

2. Interpretation of objective

The task of keeping an information system operating smoothly with a minimum of downtime is a tremendous challenge. In a complex system, like an implementation of the DII COE, careful consideration has to be given to how the network is going to be managed so that it can provide full connectivity, correct functionality, and full flexibility to the end user. The DOD is becoming increasingly reliant on information systems to

perform day to day tasks. This dependency means that there will be serious consequences for interruptions in communications in DOD systems.

An interruption or failure in the military pay system causing a delay in paydays would be a blow to morale, affecting service readiness. A failure in an intelligence system could have devastating effects. Methods to predict or rapidly detect failures and alert personnel to take remedial action can thus produce significant benefits. [HELD92, p. 2]

Thus, network operating systems and their network management tools must provide a means to monitor network equipment and facilities and provide technicians with the ability to implement configuration changes from a central site location, as well as generate alarms when predefined conditions occur. It is these capabilities that TAFIM's management objective hopes to improve.

a. Consistent management functions and interfaces

Defining a common set of management functions is essential in GCCS, or any other DII COE application. The ever changing environment in which the DII must operate, as well as constantly rotating personnel, necessitate the need for common management and functions to ensure personnel are properly and cost-effectively trained. These management functions should be common to all NOSs. The International Standards Organization (ISO) with the development of the Open System Interconnection (OSI) standards defined five network functional areas that are generally accepted by the industry and believed to be the minimum functions a network management system should include. Theses five functional areas are:

- Configuration and change management

- Fault and problem management
- Performance and growth management
- Security and access management
- Accounting and cost management

While not inclusive, these areas do provide a basic set of management functions. Administrators should be able to perform these basic management functions within an intuitive GUI, one that is consistent with the NOS.

(1) Configuration and change management

Configuration and change management involves keeping track of the many and various components of the network. It is probably the most important part of network management. Unless a network administrator can keep track of all the components of the network, he can not accurately manage the network. Configuration management software should be able to provide the network administrator with many capabilities. In a large network, control should be administrator from a single point. Administrators should be able to view graphical configurations of the network layout.

(2) Fault and problem management

Fault and problem management includes the detection, isolation, tracking, and resolution of problems which occur on the network. The most important part of this function is fault identification. This can be accomplished in a variety of ways, from setting thresholds on the network to users reporting problems encountered to the network administrator. Once problems are encountered, procedures must be established to record the problem, identify the cause, and correct it.

(3) Performance and growth management

Performance and growth management ensures that sufficient network capacity exists to support the requirements of the end user. This function evaluates the performance and use of network equipment as well as adjusting the network configuration. Typically, the evaluation could require visual observation of equipment indicators or the gathering of statistical information into a database that would be used to project trends of network use. Performance measurements provide the ability for network managers to measure network parameters such as response time, quality of service, congestion, and availability. [MULL90, p. 268]

(4) Security and access management

Security and access management includes functions which ensure that only authorized personnel access the network. Many of the functions associated with security management were discussed in the previous objective and include functions such as authentication of users, encryption of data, the management and distribution of encryption keys, examination of security logs, virus prevention measures, and the performance of audits and traces to ensure only authorized users access the network resources and facilities. [HELD92, p. 7]

(5) Accounting and cost management

Accounting and cost management functions include developing methods for establishing charges for the use of the network by various departments. Some of these functions are budgeting for resources, examining the effects of tariff

charges on the structure of the network, and verifying the correctness of vendor and telecommunication carrier bills. [HELD92, p. 8]

Another important feature for a NOS is the ease with which management capabilities are performed by the network manager. The user interface to management information, whether real time alarms and alerts or trend analysis graphs and reports, is an essential piece to successfully implementing management functions. If the data gathered cannot be transformed into useful information in an easily understood format, then the real purpose of a network management system is lost. Collecting data is meaningless if the data is not used to make informed decisions about the optimization of systems and functions. One way of achieving this objective is to have consistent, intuitive interfaces for these functions. [STEV95]

b. Management standards

One theme throughout the objectives of TAFIM is interoperability. This feature should be extended to all aspects of the network, including management capabilities. The increasing complexity and growth of networks has necessitated the development of network management tools. Complexity and growth are also likely to plague the DOD as systems are required to comply with to DII COE. Maintaining legacy systems, new acquisitions, different systems, and multiple vendor systems, complicates the methods for managing networks. In this diverse environment, managers are forced to rely on a variety of tools to keep networks optimized for efficiency and cost savings.

The need arises for standards that allow "the equipment of different vendors to interoperate on the same network, permitting the exchange of network

management information such as alarms, performance measurements, usage statistics, and diagnostic tests resulting in a standard format" [MULL90, p. 279]. Standards are needed to enable objects on a network to talk a common language, allowing them to exchange information about packets, protocols and network data. Standards in network management should provide several benefits including:

- A single network for user applications and network management that results from a common communications platform.
- Management in a multi-vendor operating environment that is facilitated by open naming conventions and standard data fields.
- Reduced costs for network management through common, reliable specifications that are incorporated into the designs of hardware and software.
- Standardized applications across network elements by common management protocol definitions. [MULL90, p. 279]

The DOD would benefit from these goals as it tries to migrate its systems to the DII COE. During this migration, the DOD will have to manage legacy systems along with new DII COE based systems. Standard management protocols would enable the DOD to more effectively achieve its objectives. As with any other standard, the difficulty is identifying which one to adopt.

The first network management standard to be developed was the Simple Network Management Protocol (SNMP). This protocol was considered a quick, short term solution while other more capable management protocols were being developed. Two management protocols that eventually emerged in the late eighties were SNMPv2 (an improved version of SNMP) and Communications Management Information Protocol (CMIP). These two were expected to succeed SNMP.

SNMP was designed to operate over TCP/IP. It was designed in the mid-1980s as an answer to managing different types of networks. As mentioned, it was first conceived of as a temporary solution until a better network management protocol became available. However, SNMP soon became widespread and the network management protocol of choice. It has become the "de facto" standard. [UWAT96]

SNMP was designed to perform the basics of network management without putting stress on network resources. SNMP operates on a network by exchanging messages that contain network information. These messages are known as protocol data units (PDU). PDUs are used by SNMP to monitor the network. They perform functions such as reading terminal data, setting terminal data, and monitoring network events like terminal start-ups.

One advantage of SNMP is that it is a simple protocol. It has smaller memory and CPU requirements than other management protocols. SNMP has been used as a management protocol on the Internet since its development, and is supported by many vendors. SNMP is in wide use, is easy to implement, and does not put stress on a network. - The shortcomings of SNMP can also be traced to its simplicity. SNMP was designed without many security features. The other complaint about SNMP is that it is so simple that it does not provide information that is detailed enough, or organized enough, to keep up with the increasing size and diversity of networks. The latest version of SNMP, SNMPv2, was designed to address these shortcomings. SNMPv2 provides more security features, as well as providing more detail for managers. Surprisingly, the original version of SNMP remains the more popular version today. Its widespread use and simplicity have kept SNMPv2 from being widely adopted. [UWAT96]

CMIP was the protocol designed to replace SNMP in the late 1980s. It was built to make up for deficiencies of SNMP. Its design is similar to SNMP, but it is a much bigger, more detailed network manager. It uses PDUs to monitor the network, but with a larger number of them. In CMIP, PDUs are much more complex and provide significant advantages. One is that in CMIP variables cannot only be used to relay information to and from a terminal line, as in SNMP. In addition, they can be used to perform tasks. In CMIP, for example, a PDU can notify the management server that a terminal can not reach its file server. In SNMP, the administrator can only determine this by keeping track of how many times a terminal has failed to respond. This makes CMIP more efficient, requiring less manpower to administer. CMIP was also built with security features in mind. It supports authorization, access control, and security logs. The result is more secure protocol. [UWAT96]

CMIP has two major disadvantages. First, it is a very large management system. It requires a large amount of system resources to administer. The requirements are ten times those of SNMP. This makes CMIP difficult to administer except on large networks. [UWAT96] Second, it is an OSI network management protocol, as opposed to a TCP/IP protocol. Since the market place has overwhelmingly opted for TCP/IP over OSI, CMIP is no longer a viable option.

The potentially large and complex information systems that could result from implementations of the DII COE, realizing that there will be legacy systems to maintain, underscores the need for a standard management protocol. It is important for the DOD to incorporate NOSs that support standard management protocols like SNMP and CMIP. These protocols will standardize the structure for formatting messages and

transmitting information between objects on a network. This will be essential as the DOD continues to develop information systems, especially if it desires to be able to manage the new systems as well as its legacy systems.

Another concern about management standards is the use of standard practices when dealing with security issues, control of accesses, and down time of the systems for network administration. This is less of a NOS issue than an issue of policies and practices.

c. Reduced costs

Ideally, the purpose of NOS management features is to bring consistency in the configuration and management of heterogeneous systems. The ability to troubleshoot problems on a network from a central location, configure and manage devices from a central location, and provide a structure for easy expansion and organization should reduce costs. [PABR96, p. 169] Increased standardization of management features would benefit DOD personnel, especially when considering frequent movement of personnel.

Traditionally, the costs of managing a network have been significant. It is estimated that firms spend on the average about 15 percent of their total information systems budget on network management. This percentage translates into an average annual expense of $1.3 million for the largest 100 American firms. [SNMP96] The ability for management features to reduce costs should be an objective when selecting NOSs, especially with today's shrinking budgets. Improved management features should promote this. This tenet will primarily be achieved by adhering to the ideas in the first two standards.

3. Analysis of the Unix architecture

Hewelett-Packard's vision of network administration as quoted from their home page is:

> HP-UX provides a core set of standard Unix network management tools. In the simplest terms, systems management is a set of procedures and tasks that are used to maintain a reliable, secure, and robust computing environment. The key to enterprise-wide IT control is a common framework. This platform, however, must support the diverse applications used by a variety of operators and administrators working in numerous physical locations. While these individuals may have specific responsibilities, they cannot work in isolation. Effective system management must link these individuals and coordinate management activities. [HPVI96]

a. Consistent management functions and interfaces

HP-UX Systems Administration Manager (SAM) provides an interface to basic Unix administrative functions. SAM enables a network administrator to perform management tasks through a GUI. This enables the administrator to run common Unix management utilities without having to remember particular commands. This simplifies tasks for the network administrator. Tasks are now accomplished through a standard management interface. SAM accomplishes this by providing a sequence of guidance steps to perform such tasks as configuring or adding a disk or adding a printer.

By simplifying complex tasks, via this set of structured questions, SAM is able to reduce errors and increase productivity. For example, to add a new disk to the system, the administrator simply selects this task from the SAM interface and is prompted for the directory on which to mount the disk. All other auto-sensing and configuration tasks are handled transparently. TCP/IP and links can also be configured through SAM.

Another benefit that SAM provides is the ability to be customized. SAM is capable of allowing third party tools and utilities to be launched through SAM's graphical interface, keeping the interface to the administrator consistent. This enables all management utilities to be maintained and administered through the same interface. [HPVI96]

SAM is also highly scaleable. SAM accomplishes scalability by creating a single superuser. This superuser can assign other administrators privileges to accomplish a portion of the task that a superuser is responsible for. This delegation of tasks allows the superuser to be in charge of the system, while at the same time allowing many tasks to be spread out to reduce the burden on the superuser. [HPVI96]

HP-UX also includes a standards based utility for managing software. This utility, called Software Distributor/UX, is based on the draft copy of POSIX 1387.2. Software Distributor/UX offers the administrator the means to distribute, manage, and install from one location. Software Distributor/UX is also run from the same SAM interface. [HPVI96]

All of the HP-UX management tasks described are based on an object/task design. This simplifies network management for the administrator. All hardware and software components are treated as objects; these objects perform tasks. This level of abstraction simplifies the job of the system administrator by using consistent methodology to manage different elements in the network environment. Additionally, all the tools mentioned have the identical graphical interface. This allows the administrator to learn only one interface to perform all management tasks.

b. Management standards

Unix NOSs have provided support for TCP/IP since the University of California, Berkeley added it to their variant in 1981. This support lends itself well to the Internet, but it also provides easy scalability. The network administrator can assign each sub-net, usually a LAN, a set of addresses. TCP/IP, a router-based protocol, can use routers and bridges between sub-networks to prevent congestion and keep resources available to users by isolating traffic.

Not only does HP-UX support the TCP/IP protocol stack, it also supports the TCP/IP based management protocol SNMP. SNMP provides HP-UX flexible management capabilities. This allows a Unix computer to manage objects remotely using SNMP based management software. As discussed, SNMP is the most prevalent network management protocol.

4. Analysis the Windows NT architecture

Microsoft stresses that system administration tools within Windows NT differ from traditional network administration tools because of two factors. The first is that Windows NT is based on a client/server model rather than a traditional host based network model. This makes it possible to use decentralized administration. However, most organizations would chose centralized administration networks because of the greater control it provides. The second reason is that all administration in Windows NT is performed from the traditional Windows GUI. This allows administrators to conveniently use tools that accomplish administrative functions.

a. Consistent management functions and interfaces

Administrative tools that are included within Windows NT include:

- Performance Monitor
- Event Viewer
- Server Manager
- User Manager for Domain
- Disk Administrator

(1) Performance monitor

Performance Monitor is the Windows NT tool for tracking performance. The Performance Monitor tracks network parameters such as: the number of processes waiting for disk time, the number of network packets transmitted per second, and the percentage of processor utilization. All information is displayed graphically, but can also be given in text format. Data can be displayed either in real time or collected in logs for later use. Windows NT Performance Monitor can also be used to generate alert logs. Alert log entries can be made at times when specified limits are exceeded on the network. [MICR03, p. 34]

(2) Event viewer

Windows NT Event Viewer can track a range of events that occur on a network, from system wide events to events initiated by a single user. System wide audit policies are established by administrators through User Manager for Domains. Windows NT uses three types of logs to record events that occur on the network. These logs can be viewed by the network administrator through the event viewer. The System

Log tracks events triggered by Windows NT components. Examples include components that fail to load during startup and power fluctuations. The Security Log tracks events triggered by security violations. This would include illegal logons and unauthorized file access. The Application Log tracks events triggered by application programs. The Event Viewer enables administrators to examine and manipulate log entries.

(3) Server manager

The Server Manager in Windows NT is an administrative tool used to perform a number of tasks. Users can be inspected from within Server Manager. Messages concerning network status can be transmitted using Server Manager. With Server Manager, administrators are capable of inspecting logged-in user accounts, shared resources, connections, replication, and administration alerts. The network administrator can also determine which resources are currently being used, or even who is connected to a resource and duration of use.

(4) User manager

The User Manager for Domains function in Windows NT is used to create and modify user profiles. Figure 17 shows how the network administrator can control access to servers and workstations within a domain. [MICR03, p. 32] User Manager also allows the administrator to restrict hours of use, valid dates that a user can access the network, and which resources the user can access.

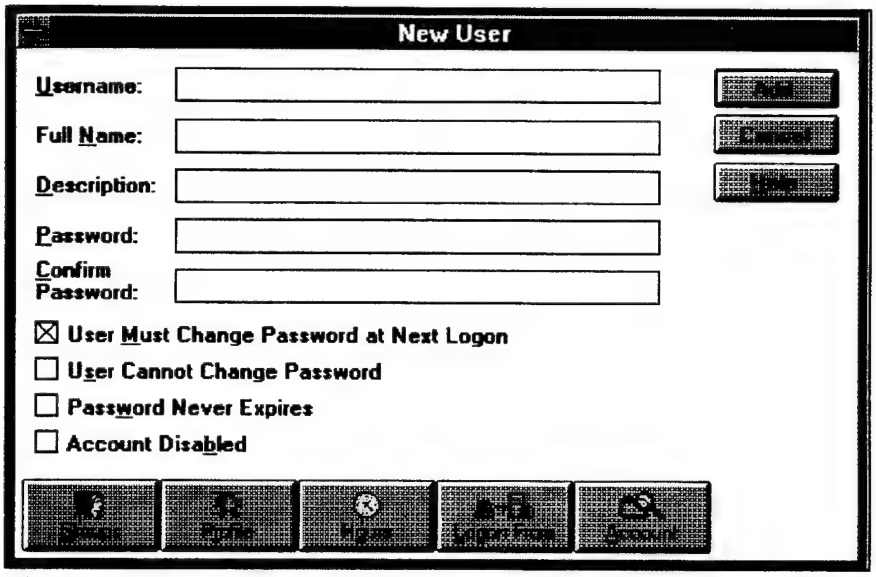

Figure 17. User Manager for Domains - New User Dialog Box

(5) Disk administrator

Disk Administrator is a utility which allows the administrator to perform virtually any task involving disk drives. These tasks include: creating partitions, creating volumes and stripe sets, reading status information (e.g., partition size, block size, etc.), and assigning drive letters to partitions.

Disk Administrator also includes several additional features. One of these capabilities is to statically assign drive letters. This is so the order of existing drive letter assignments will not be disturbed. Another feature is the ability to search for information such as assigned drive letter or stripe set, which is very useful when installing a new copy of Windows NT. [RULE95, p. 127]

Due to the fact that all the management tools mentioned above are built into Windows NT, all of the functions are convenient to use. This makes the job of the administrators easier because they do not have to learn third party tools.

b. Management standards

By standardizing on Windows NT, a familiar interface between the administrator and the NOS is achieved. Windows NT allows the use of several different platforms and protocols on the same network using the same set of resources such as servers, printers, modems, and scanners. By supporting these different platforms and protocols, management needs only one set of management tools for the entire network, instead of a different set of tools for each type of protocol or platform on the network.

Because Windows NT includes TCP/IP, a protocol available in many NOSs, Windows NT has the capability of communicating with different NOSs. Windows NT also supports SNMP. This allows a Windows NT computer to be managed remotely by SNMP-based management software. [RULE95, p. 531]

5. Summary of findings

Both HP-UX and Windows NT provide basic administrative functions to help network administrators. Both Microsoft and Hewlett-Packard admit that a much more robust management software package is needed for managing large distributed networks. Both in fact have developed add-on management packages that can be purchased by the customer.

Hewlett-Packard has developed a strategy to incorporate systems and network management applications under a common framework. This framework, HP OpenView, provides a strategy for managing multi-vendor networks, systems, applications, and databases across both legacy and client/server systems.

Microsoft has similarly developed an additional management package called Systems Management Server. This product is an add-on product contained in the product BackOffice for Windows NT. However, it does not contain all the tools necessary for network management, which are contained in other products based on SNMP (e.g., Hewlett-Packard's OpenView) [RULE95, p. 394]. Microsoft acknowledges Windows NT's lack of complete administrative support for large networks by stating:

> For more sophisticated tools, the kind of tools you need for data center operations, there are third-party products available for Windows NT. (Some of them have even been ported from Unix). [MICR03, p. 37]

The difficulty then is trying to determine which NOS comes closer to meeting the objectives of TAFIM, realizing that neither provides enough network management capabilities without add-on software. The authors do feel that Windows NT fails to provide adequate TCP/IP support. The most dramatic failures are that Windows NT does not support dynamic routing, or Telnet server. John D. Ruley, editor at large of Windows Magazine states, "For complex networks with multiple paths to the same remote destination, you must use a non-Windows NT system that supports dynamic routing." [RULE95, p. 328]. Windows NT static routing model, which requires manual configuration, suffers performance degradation after about five segments. Manual administration becomes overwhelming at this point. [RULE95, p. 329] The demands of the DII COE require the capability of having multiple paths to critical locations. Almost all Unix variants, including HP-UX, support RIP or other dynamic routing protocols contained in the TCP/IP protocol suite. [RULE95, p. 331] Because Unix supports dynamic routing, it can more easily manage a complex network. The Telnet server

deficiency means that an organization must install a Unix Telnet server in order to provide this service.

Unix relies more on management standards, like SNMP and TCP/IP. This reduces the costs of using third party products. For these reasons, the authors feel that although not perfect, Unix meets this TAFIM objective more completely than Windows NT.

IV. CONCLUSION

A. DETERMINING THE RIGHT NOS

1. Which NOS is better for the DII COE?

A summary of the analysis contained in Chapter III is included in a matrix in Appendix E. The matrix shows how each NOS measures up to each TAFIM objective and the objective's corresponding tenets. The summary clearly shows that each NOS has its advantages and disadvantages. The authors feel that neither NOS adequately meets all the DII COE needs and all the guidelines of the TAFIM TRM objectives. Appendix D provides additional information about the Unix and Windows NT NOSs for information managers.

Unix and Windows NT are comparable in many key areas: 32 bit support, multitasking, multithreading, security, integrated networking, and support for symmetric multiprocessing. The advantages of Unix include its maturity, *open* systems standards support, distributed networking, support for parallel processing, and scalability. Windows NT's advantages include a more familiar user interface, OLE support, reduced training costs, portability, and support for a large number of Windows applications. This thesis concludes that the DOD should capitalize on the strengths of both NOSs by matching each NOS to those tasks where it can provide better support. Unix and Windows NT would best serve the DOD by being used only for those tasks where each NOS's strengths will benefit the end users.

DOD information systems that successfully employ Unix servers, with its high end scalability on large information servers, and Windows NT on the desktop, with its lower cost and user friendly interface, maximize the capabilities of the information systems and their NOSs. The employment of both Unix and Windows NT requires the two NOSs to interoperate to a high degree. The authors feel that the technology and functionality of both NOSs are adequate to support this level of interoperability.

By limiting an information system to one of the two NOSs, the usability and performance of the information system would be less than the minimal needs as outlined in TAFIM. Both NOSs combined together to utilize strengths and minimize weaknesses, still do not provide all the features required to fulfill all the needs of DOD information systems, let alone TAFIM.

Nevertheless, real world needs demand that information technology managers select a NOS for their information system. This selection must be made to meet the TAFIM objectives as best as possible. Today's market place does not permit the use of a non-proprietary NOS, because one simply does not exist. In order to be functional, DOD units must own and maintain a computer information system, and therefore a NOS must be used. One of the goals of the DII COE is to reduce the number of different NOSs used by the DOD. A combination of Unix and Windows NT is certainly not ideal, nor does it meet all of TAFIM's objectives, but no single NOS is enough.

2. Where are we?

The intent of TAFIM is to provide DOD information managers guidelines for developing standard information architectures. This document is not intended to provide an architecture for a specific DOD mission. Consequently, it defines a set of ambitious

objectives. Several of these objectives contained in the TAFIM TRM are vague, abstract, and redundant. Some are, for the most part, unattainable with current technology in the market place. Some objectives (e.g., promote vendor independence and improve development efficiency) are lofty and not necessarily beneficial to the DOD. Specifically pertaining to NOSs, TAFIM's large size and general approach leave information managers lacking enough specific guidance to help select an appropriate NOS. TAFIM was originally designed around the Unix operating environment because, at the time of TAFIM's writing, Unix was the only real NOS that the DOD could employ to handle its information systems. Consequently, many of the objectives and standards that the TAFIM TRM adopts bear a legacy to Unix. DOD system developers need to approach TAFIM as only a guideline to designing standard architectures, not one that suggests a specific architecture, like Unix.

The DII COE takes TAFIM one step further toward providing the DOD with a specific system architecture. While the DII COE does provide more specific guidance to system designers, this document is also written in general terms. This leaves information managers alone to decide specific system questions like what NOS to use.

B. REMAINING INFLUENCING ISSUES

1. Where should we be going?

While not perfect, it is important to realize the benefits and drawbacks of having the DOD adopt the TAFIM framework. Although TAFIM may describe a set of ideals that are difficult to obtain, it does present DOD system developers with a common set of objectives to strive for. Interoperability, portability, scalability, and reducing life cycle

costs will continue to be important issues when designing systems. Both TAFIM and the DII COE emphasize DOD adoption of *open* systems standards in order to eliminate reliance on proprietary vendors, and increase interoperability and portability. TAFIM and the DII COE should become continuously updated living documents in order to increase manageability and decrease redundancy. There should be one framework that outlines general objectives and another one that addresses specific factors and variables to direct information system managers. The guidance should not be the adoption of *open* systems standards, but a monitoring of the market place for specific direction.

2. Why did we choose TAFIM objectives?

Since the environment for this thesis was the DII COE, and the DII COE states it is fully TAFIM compliant, the TAFIM TRM was chosen as the framework to use for discussion and comparison. The assumption was made that this framework provided sufficient guidance in analyzing NOSs. The question is whether TAFIM alone is a sufficient framework for analyzing how well a NOS meets the needs of DOD information systems. The authors conclude that an information system framework can provide guidance to decision makers in developing or choosing a NOS. The problem with TAFIM is that it is too general in nature and does not provide any specific guidance to information managers.

TAFIM fails to mention or address certain fundamental objectives crucial to NOS selection such as network performance, fault tolerance of critical systems, and reliability. If TAFIM is going to be the document that the DOD uses to design system architectures, it needs to become a living document that is updated and modified to fit current needs and objectives of all system aspects, including NOSs. It is the opinion of the authors that

TAFIM is not specific enough, and can not therefore provide sufficient guidance to information system mangers in its present state.

3. What have we accomplished?

This thesis provides guidance to decision makers in picking a NOS that is best for their specific information system. Commanding officers and system administrators throughout the DOD still face the difficult decision of deciding which NOS is best for their information system. It is not feasible to provide case by case guidance for each information system in the DOD. This thesis provides guidance and a framework for those decision makers to use when evaluating their specific systems, and deciding how they should choose the NOS for it.

Windows NT is better for desktop computing and mid-level (departmental) servers while Unix is better for high end servers. The term *better* here does not indicate that all functionality and services required are provided, it simply means *better* relative to the other NOS. DOD decision makers must realize that either NOS without add-on third party products will not necessarily fulfill all their needs, and will not satisfy all the objectives outlined in TAFIM. To date, such a NOS does not exist on the market. The perplexing issue that still remains is whether both Unix and Windows NT will continue to evolve and eventually meet more of the objectives outlined in TAFIM.

4. How can you achieve added benefits for your information system?

This thesis analyzed the capabilities provided by the Windows NT and Unix NOS. Both NOSs failed to fulfill several of the tenets contained within each TAFIM objective. Many of these tenets, however, are attainable by adding third party solutions to each NOS.

The authors deliberately tried not to discuss add-on third party products or options. Doing so would have dramatically increased the scope of the thesis as well as reduced apparent differences between the two NOSs. Ideally, when purchasing a NOS, the DOD would obtain a product that fulfilled required functionality of DOD information systems. However, the commercial market does not need the same things that the DOD requires in a NOS, and therefore the market place is not likely to fulfill DOD needs anytime soon.

The best option today for DOD information systems is to adopt one or both of these NOSs and tailor them to fulfill specific organizational needs with third party products or by custom development. Minimal research can yield many market solutions to most of the needs not covered either by the Windows NT or Unix NOS.

5. What is the conclusion on standards?

Open systems standards are prominent throughout all the objectives of TAFIM. How does the DOD adopt a proprietary NOS that fulfills its needs and still maintain vendor independence? The push by the federal government to outsource is easing the tension in this area, but it is still a concern to decision makers. However, by choosing between either the two approved Unix variants or Windows NT, the DOD is essentially in that position. Although it is arguable that Unix is less vendor dependent that Windows NT, dependencies still exist and will continue to be an issue that needs to be addressed.

The development of interoperable heterogeneous information systems, through standardization, continues to be a focal point in the commercial market place. The authors would like to conclude that the DOD, although a major force, should not be in the business of developing these standards. DOD should let the market choose the path of standards and follow these current trends in the market place.

C. AREAS FOR FURTHER RESEARCH

Areas of research that can help DOD information system decision makers are to address some of the most effective third party add-on products and services. For example, this can be accomplished by taking an objective view of products such as Microsoft BackOffice (and its numerous applications) for system administration or HP OpenView for providing better security and administration on Unix platforms.

Another area of research that can be explored is the update and modifications required to TAFIM in order to make it an up-to-date framework for information management. Determining which objectives no longer apply today and which objectives have been changed with the advancement of new technology are just the beginning for further research into TAFIM.

While it is important to study reducing life cycle costs, the DOD would benefit from a thorough cost/benefit analysis when deciding potential NOSs. This is worthwhile only after the determination of a viable solution. All things equal, in today's fiscally sensitive military, there is a need to select the most cost efficient solution. If there is only one solution, then matters are simplified a great deal.

APPENDIX A. GCCS COE AS-BUILT STANDARDS

The GCCS COE is being implemented primarily through the integration of existing components provided by the Services and Agencies. Limited new development is taking place to add additional functions, port to the required platforms, allow multiple languages to call COE components, and aid in integration. Thus, the standards profile primarily represents an as-built documentation of what exists. The as-built standards meet three tests: there is a direct or derived mission area application need, the standard is mature, and products are available, if necessary, to implement the standard. Other standards have been included in the GCCS standards profile to support future development.

This profile does not include the specification of the project APIs that have been adopted by GCCS. Project APIs document the interface to developed software that has been included in the GCCS COE. These APIs will be documented under separate cover. Table 3 describes each standard and it's application in the GCCS COE.

Service Area	Service	Standard
Operating System	Kernel Shell & Utilities Real-time Extension	POSIX compliant Unix
Software Engineering Services	Programming Languages Case Tools & Environment	FIPS PUB 119 (Ada) FIPS PUB 119-1 (Ada-9X) FIPS PUB 160 (C) Developer & Service Specific
User Interface	Client/Server Object Definition and Management Window Management	FIPS PUB 158 (X-Window) MOTIF DOD Human Computer Interface Style Guide MOTIF FIPS PUB 158 (X-Window) MOTIF
Data Management	Data Dictionary - Directory Data Management	FIPS PUB 156 (IRDS) FIPS PUB 127-1 (SQL) FIPS PUB 127-2 (SQL+)
Data Interchange	Document Interchange Vector Graphics Data DMA Vector Map Data Raster Graphics Data DMA Raster Data Symbology Imagery DBMS data	WP 5.1, ASCII FIPS PUB 128 (CGM) VPF/SDTS/DTED/DCW FIPS PUB 150 (Type I) ADRG/ADRI NATO STANAG 2019 SMGS NITF IDBTF, IDBEF
Graphics	Graphics Map Products	FIPS PUB 120-1 (GKS) FIPS PUB 153 (PHIGS) DNC, DTED, ARC, VMAP, World Databank II, CARDG, World Vector Shoreline
Network	Data Communications	FTAM/X.400/X.500/X.25 IEEE 802.2 802.3, 802.4,

Service Area	Service	Standard
	PC/Micro Support	802.5 FTP/TELNET/SMTP/TCP/IP, MIL-STD-187-700 FIPS 160-170 (Modem Support)
Security	Evaluation Criteria Operating System Data Management Network Services	DOD 5200.28-STD CSC-STD-003-85 IEEE P1003.6 NCSC-TG-021 (TDI) NCSC-TG-005 (TNI)
Distributed Computing GCCS COE As-Built Standards Profile	Distributed Data Transparent File Access Distributed Computing	None NFS RPC, Berkley Sockets

Table 3. GCCS COE As-Built Standards Profile

APPENDIX B. COMMON CONSENSUS STANDARDS

Service Area	Service	Standard
Operating System	Kernel	FIPS Pub 151-1 (POSIX.1)*
	Shell and Utilities	IEEE P1003.2*
	Realtime Extension	IEEE P1003.4*
Software Engineering Services	Programming Languages	FIPS PUB 119 (ADA)*
	Case Tools and Environment	ECMA Portable Common Tool Environment (PCTE) Specification 149
User Interface	Client Server Operations	FIPS PUB 158 (X-Window System)*
	Object Definition and Management	DoD Human Computer Interface Style Guide*
	Window Management	FIPS PUB 158 (X-Window System)*
	Dialogue Support	Future Standard IEEE P1201.X*
Data Management	Data Dictionary - Directory	FIPS PUB 156 (IRDS)*
	Data Management	FIPS PUB 127-1 (SQL)*
Data Interchange	Document Interchange	Planned FIPS PUB (Office Document Architecture/Office Document Interchange Format/Office Document Language ODA/ODIF/ODL)*
	Document Interchange	FIPS PUB 152 (SGML)*
	Vector Graphics Data	FIPS PUB 128 (CGM)*
	Raster Graphics Data	FIPS PUB 150 (Type I)* Planned FIPS PUB (Type II)*
	Product Data Interchange	Planned FIPS PUB (Initial Graphic Exchange Specification/IGES)*
	Product Data Interchange	Draft International Standard (Standard for the Exchange of product Model Data-STEP)
	Electronic Data Interchange	FIPS PUB 161 (EDI)*
Graphics	Graphics	FIPS PUB 120-1 (GKS)*
	Graphics	FIPS PUB 153 (PHIGS)*

Network	Data Communications	FIPS PUB 146-1 (GOSIP)*
	Telecommunications	MIL-STD-187-700*
Security	Evaluation Criteria	DOD 5200.28-STD*
	Compartmented Mode Workstation	DRS-2600-5202-87*
	Compartmented Mode Workstation Evaluation Criteria	DRS-2600-6243-91, Version 1*
	Compartmented Mode Workstation Labeling: Encoding Format	DDS-2600-6216-91*
	Compartmented Mode Workstation Labeling: Source Code and User Interface Guidelines	DDS-2600-6215-91*
	Digital Signature	Draft FIPS PUB (DSS)*
	Operating System	IEEE P1003.6 (Draft Standard)*
	Data Management	NCSC-TG-021 (TNI)*
	Network Services	ISO 7498-2*
	Network Services	NCSC-TG-005 (TNI)*
	Network Services	Draft IEEE Standard 802-10*
	Network Services	DNSIX; Version 2.1
	Network Services	Draft ISO Standard for Transport Layer Security Protocol (TLSP)*
	Network Services	ISO Committee Draft for Network Layer Security Protocol (NSLP)*
Distributed Computing	Distributed Data	ISO 9579 -1, 2 Remote Database Access (RDA)*
	Transparent File Access	Draft IEEE Standard P1003.8
	Distributed Computing	Draft OSF Specification (NCS/RPC)
System Management	System Management	Government Network Management Profile (GNMP) FIPS 179*
Internationalization		

Table 4. Summary of Consensus Standards from [DISA03, Vol 2 p. 3-5]

Table Explanation:

Entries with an "*" indicate standards within the DoD Profile of Standards.
Entries in shaded areas are under consideration and are for planning purposes only.

APPENDIX C. ORANGE BOOK CLASSIFICATIONS

Feature	D	C1	C2	B1	B2	B3	A1
SECURITY POLICY							
Discretionary Access Control	-	X	X	S	S	X	S
Object Reuse	-	-	X	S	S	S	S
Labels	-	-	-	X	X	S	S
Label Integrity	-	-	-	X	S	S	S
Exporting Information	-	-	-	X	S	S	S
Labeling of Output	-	-	-	X	S	S	S
Mandatory Access Control	-	-	-	X	X	S	S
Subject Sensitivity Labels	-	-	-	-	X	S	S
Device Labels	-	-	-	-	X	S	S
ACCOUNTABILITY							
Identification and Authentication	-	X	X	X	S	S	S
Audit	-	-	X	X	X	X	S
Trusted Path	-	-	-	-	X	X	S
ASSURANCE							
System Architecture	-	X	X	X	X	X	S
System Integrity	-	X	S	S	S	S	S
Security Testing	-	X	X	X	X	X	X
Design Specification & Verification	-	-	-	X	X	X	X
Covert Channel Analysis	-	-	-	-	X	X	X
Trusted Facility Management	-	-	-	-	X	X	X
Configuration Management	-	-	-	-	X	S	X
Trusted Recovery	-	-	-	-	-	X	S
Trusted Distribution	-	-	-	-	-	-	X
DOCUMENTATION							
User's Guide to Security	-	X	S	S	S	S	S
Facility Security Manual	-	X	X	X	X	X	S
Test Documentation	-	X	S	S	X	S	X
Design Documentation	-	X	S	X	X	X	X

X = New requirements for this class

S = Requirements are the same as the previous level

APPENDIX D. UNIX AND WINDOWS NT AT A GLANCE

NOS Feature	Unix (HP-UX)	Windows NT
System		
System architecture	32 Bit	32 Bit
Multi-platform support (portability)	Yes	Yes
File system	NFS	NTFS
Multitasking support	Yes	Yes
Distributed computing support	Yes	Yes
Symmetric multiprocessing capable	Yes	Yes
Max. processors per machine	14	32
Parallel processing capable	Yes	No
Support for POSIX 1003.1	Yes	Yes
Max addressable RAM	3.75 Gigabits	4 Gigabits
Max. file size	128 Gigabits	Several Exabits (2^{64} bytes)
File recovery support	Yes	Yes
Logical volume management for large disks	Yes	Yes
Disk mirroring support	Yes	Yes
RAID support	Yes	Yes
OLE support	No[12]	Yes
Remote procedure call support	Yes	Yes
Max. media storage size	Virtually unlimited	17 Billion Gigabits
Network		
Network manageability support	OpenView	Backoffice
SNMP support	Yes	Yes

[12] Microsoft offers a Windows Interface Source Environment add-on to Unix which provides OLE support on Unix systems.

NOS Feature	Unix (HP-UX)	Windows NT
Dynamic Host Configuration Protocol support (DHCP)	Yes	Some
Type of network system	Host-based	Client/Server Peer-to-Peer
Supports multiple network protocols	Optional	Yes
User Interface		
Support for X-Windows/Motif	Yes	Yes
Support for Win32 GUI	No	Yes
Common desktop environment support	Yes	No
Security		
Boot authentication	Yes	No
User log-on required	Yes	Yes
Per process memory protection	Yes	Yes
File-level access permissions	Yes	Yes[13]
File-access control lists	Yes	Yes
Security auditing	Yes	Yes

[13] Windows NT and Unix both offer read, write, and execute permissions on each file. Windows NT adds 'take ownership' and 'change permission' to the common set of file attributes.

APPENDIX E. UNIX AND WINDOWS NT COMPARISON MATRIX

TAFIM TRM Objectives and Tenets	Unix	Windows NT
Objective 1: Improve User Productivity	√	√+
Consistent User Interface	√	√+
Integrated Applications	√	√
Data Sharing	√	√+
Objective 2: Improve Development Efficiency	√	√
Common Development	√	√
Common Open Systems Environment	√	√
Use of Products	√	√
Software Reuse	√	√
Resource Sharing	**N/A**	**N/A**
Objective 3: Improve Portability and Scalability	√	√
Portability	√	√+
Scalability	√+	√
Objective 4: Improve Interoperability	√	√
Common Infrastructure	√	√
Standardization	√	√
Objective 5: Promote Vendor Independence	√+	⇩
Interchangeable Components	√+	√
Non-Proprietary Specifications	√	⇩

TAFIM TRM Objectives and Tenets	Unix	Windows NT
Objective 6: Reduce Life-Cycle Costs	√	√
Reduced Duplication	√	√
Reduced Software Maintenance Costs	√	√
Reduced Training Costs	√	√
Objective 7: Improve Security	⇓	⇓
Uniform Security Accreditation and Certification	√+	√
Consistent Security Interfaces	√	√+
Support for Simultaneous Processing in Single Platforms of Different Information Domains	⇓	⇓
Support for Simultaneous Processing in a Distributed System of Different Information Domains	⇓	⇓
Support for Use of Common Use: Communications Systems	⇓	⇓
Objective 8: Improve Manageability	√	√+
Consistent Management Interface	√	√+
Management Standardization	√	√
Reduced Operations, Administration, and Maintenance (OA&M) Costs	√	√

Legend:

√ = Achieves the objective or tenet

√+ = Achieves the objective or tenet better than the other NOS

⇓ = Fails to meet the objective or tenet

LIST OF REFERENCES

[AETC01] Air Education and Training Command, *What is GCCS?*, Briefing slides describing the Global Command and Control System.

[AIMT95] AIM Technology, Unix System Price Performance Guide, AIM Technology, 1995.

[ANDL90] Andleigh, Prabhat K., Unix System Architecture, Prentice Hall, 1990.

[ANSI96] American National Standards Institute Web page, available on the Internet at http://www.ansi.org/broch1.html, 25 August 1996.

[ARNO93] Arnold, Derek N., Unix Security: A practical Tutorial, McGraw-Hill Inc., 1993.

[AWUA95] Awuah, Patrick and Lazar, David, *Microsoft Windows NT Server 3.5 Remote Access Service (RAS)*, Microsoft White Paper, 1995.

[BACH86] Bach, Maurice J., The Design of the Unix Operating System, Prentice Hall, Inc., 1986.

[BARA93] Baran, Nicolas, *Windows NT supports POSIX, but does it matter?*, Byte Magazine, November 1993.

[BERT87] Bertsekas, D. and Gallager, R., Data Networks, Prentice Hall, Englewood Cliffs, NJ, 1987.

[BUTL96] Butler, Shawn, Defense Information Services Agency, *DII Common Operating Environment (COE)*, DISA briefing slides, 24 April 1996.

[CANC95] Gancarz, Mike, The UNIX Philosophy, Digital Press, Boston, 1995.

[COMP96] Computer News Daily, available on the Internet at http://nytsyn.com/live/News3/213_073196_24839.html.

[CONS01] Constance, Paul, *Buying secure products? Look closely at vendor claims.*, Government Computer News, 4 September 1995.

[CONS02] Constance, Paul *C2 rating aside, NT isn't secure.*, Government Computer News, 4 September 1995.

[CONS03] Constance, Paul, *New Joint OPS System will be Built with Reused Code*, Government Computer News, 5 June 1995.

[CS2970] Stemp, Roger, *CS2970 Class Notes*, Naval Postgraduate School, 7th quarter 1996.

[CUMM96] Cummings, Anne, Hewlett-Packard, Customer Service Representative for government contracts, Phone conversation conducted on 3 September 1996.

[CUST93] Custer, Helen, Inside Windows NT, Microsoft Press, 1993.

[DISA01] Defense Information Systems Agency, *Defense Information System Network Telecommunication Network and Strategy*. available on the Internet at http://www.disa.mil/info/pao04l.html.

[DISA02] Defense Information Systems Agency, *Defense Information Infrastructure Master Plan, Executive Summary*, available on the Internet at http://www.disa.mil/dii/diiexe/execsum1.html.

[DISA03] Defense Information System Agency Center for Architecture, *Department of Defense Technical Architecture Framework for Information Management series, Version 2.0*, DISA, 30 June 1994.

[DISA04] Kurkowski, Stuart, *Defense Information Infrastructure Common Operating Environment Integration and Run Time Specification DII COE I&RTS*, Defense Information System Agency, 13 February 1996.

[DISA05] Shroeder, Doug, *COTS & GOTS Supporting Software for GCCS Version 2.1*, facsimile, 1996.

[DISA06] Defense Information Systems Agency, *GCCS Baseline Common Operating Environment*, DISA, 28 November 1994.

[DTKS96] The Desktop Korn Shell Internet page available 1 September 1996 at http://www.unx.com/~pend/dtksh.html#whatiscde.

[DUNP94] Dunphy, Ed., The Unix Industry and Open Systems in Transition: A Guidebook for Managing Change, John Wiley & Sons, Inc. 1994.

[EETI96] EE Times Online - available on the Internet at http://techweb.cmp.com/eet/current/hr.html#dataquest.

[FEIB95] Feibel, Werner. Novel's Complete Encyclopedia of Networking, Novel Press, San Jose, CA, 1995.

[FONG94] Fong, T., *Global Command and Control System (GCCS) JIEO Engineering Plan*, DIS, 1994.

[GASK95] Gaskin, James E., The Complete Guide to Netware 3.1, Sybex Inc., 1995.

[GAUS93] Gauss, J. A., RADM, USN, *Navy Command Systems Strategy for the 1990's*, Briefing, 07 December 1993.

[GAUS96] Interview with Admiral Gauss, *He's Helping DOD's Parts Fit Together*, Government Computer News, 4 March 96.

[GCNP96] Silver, Jutdith, *Apple is the smoothest feds operator*, Government Computer News, 8 January 1996.

[HALF96] Halfhill, Tom R., *Unix Versus Windows NT: Microsoft's Flagship OS hasn't overtaken Unix, but savvy system managers are definitely taking Windows NT more seriously*, Byte Magazine, p. 42-52, May 1996.

[HANT96] Lee Ann Hantula of Anderson Consulting, Quote via e-mail, September 1996.

[HARC96] Harchelroad, Major Joan L., *Defense Information Infrastructure (DII) Common Operating Environment (COE) Configuration Management Process*, DISA briefing, 14 February 1996.

[HELD92] Held, Gilbert., Network Management: Techniques, Tools and Systems, John Wiley and Sons, New York, 1992.

[HEWP95] HP Product Briefing, *HP 9000 C-Class Workstation Models C100 and C110*, Hewlett-Packard Co. 1995.

[HPAC96] Hewlett-Packard TAC-4 Internet page, available on the Internet at http:// www.hp.com/go/tac4.

[HPVI96] *HP version 10 operating system,* Hewlett-Packard, available on the Internet at http://www-dmo.external.hp.com:80/gsy/software/hpux10.html.

[HUDG96] Hudgins-Bonafield, Christine, *The H-Report: Which Operating System for your 'Intranet'?*, Network Computing Online, 15 January 1996.

[INTE95] Char, Orrin; Evans, Cindy; and Bisbee, Robert, *Operating System Scalability: Windows NT vs. Unix*, Intergraph Corporation, 31 July 1995.

[ISOR96] International Standards Organization WEB page, available on the Internet at http:// www.iso.ch/infoe/intro.html, 25 August 1996.

[JCSP92] Joint Chiefs of Staff, *C4I for the Warrior*, 1992.

[JOKE94] Jo, Kenneth Y., McGreer, Michael M., and O'Brien, Gregory J., *GCCS Target Architecture Modeling*, IEEE, 1994.

[KAMA94] Kamat, Hrishi., *The Advantages of Open Systems*, American City & County, March 1994.

[KING94] King, Adrian, Inside Windows 95, Microsoft Press, 1994.

[KUHN91] Kuhn, D. Richard, *IEEE's POSIX: Making Progress*, IEEE Spectrum, December 1991.

[LIEN80] Lientz, Bennet P. & Swanso, E. Burton, Software Maintenance Management: A study of the Maintenance of Computer Application Software in 487 Data Processing Organizations, Addison-Wesley Publishing Company, Inc., 1980.

[MACD95] MacDonald, Dave, *Microsoft Windows NT 3.5/3.51: TCP/IP Implementation Details*, Microsoft White Paper, Microsoft, September 1995.

[MALA92] Malamud, Carl, *Analyzing Sun Networks*, Van Nostrand Reinhold, 1992.

[MASU95] Masud, Sam, *DISA Will See the Big Picture*, Government Computer News, 5 June 1995.

[MCCA95] McCarthy, Shawn P., *NT is POSIX-Complaint, GSBA Decides; Ruling Raises Questions*, Government Computer News, 21 August 1995.

[MCCA96] McCarty, Shawn P. *NSA will pay for B-level NT development*, Government Computer News, 12 January 1996.

[MFAQ96] Motif frequently asked question Internet page available at ftp://ftp.cen.com/pub/Motif-FAQ.

[MICR01] Microsoft, *Microsoft Windows NT Server 3.5/3.51: Domain Planning for your Enterprise*, Microsoft White Paper, 1995.

[MICR02] Microsoft, *The Microsoft Strategy for Distributed Computing and DCE Services*, Microsoft White Paper, 1995.

[MICR03] Microsoft, *Microsoft Windows NT From a Unix Point of View*, Microsoft White Paper, 1995.

[MICR04] Microsoft, *Reliability and Fault Tolerance*, Microsoft, 1995.

[MICR05] Microsoft, *Scalability*, Microsoft, 1995.

[MICR06] Microsoft, *Security*, Microsoft, 1995.

[MICR07] Microsoft, *The Windows NT Workstation 3.51 Advantage*, Microsoft Internet White Paper, available on the Internet at http://www.microsoft.com/NTWorkstation/NTWarp.htm.

[MICR08] Microsoft, *Windows NT Platform Gets C2 Evaluation*, Microsoft Internet White Paper available on the Internet at http://www.microsoft.com/NTWorkstation/C2.htm#choice.

[MICR09] Microsoft Corporation, *Microsoft Windows NT Workstation Installation Guide*, Microsoft Press.

[MICR10] Microsoft Corporation, *Windows NT File System: Built for Data Security*, available on the Internet at http://www.microsoft.com/NTserver/ntfs_mb.htm.

[MICR11] Microsoft Corporation, *Windows NT Workstation 3.51 Product Overview*, available on the Internet at http://www.microsoft.com/NTWorkstation/ntw351c.htm.

[MICR12] Microsoft Corporation, *Windows NT Platform Gets C2 Evaluation*, available on the Internet at http://www.microsoft.com/NTWorkstation/c2.htm.

[MICR13] Microsoft Corporation, *Security of Windows NT Server*, available on the Internet at http://www.microsoft.com/ntserver/ttsecur.htm.

[MICR14] Microsoft Corporation, *Integration of Windows family--based and UNIX Systems*, available on the Internet at http://www.microsoft.com/technet/boes/bo/winntas/technote/nt204.htm#integration.

[MOTI96] MOTIF, *How to get OSF*, available on the Internet at http://www.rahul.net/kenton/GettingMotif.html.

[MULL90] Nathan P. Muller and Robert P. Davidson, LANS to WANS: Network Management in the 1990s, Artech House, Boston, 1990.

[NMSG95] Joint Chiefs of Staff, *National Military Strategy of the United States of America*, Chairmen of the Joint Chiefs of Staff, February 1995.

[NORT91] Norton, Peter and Hahn, Harley, Peter Norton's Guide to Unix, Bantam Books, 1991.

[NRAD01] Navy Research and Development Command and Intelligence Systems Division, Description of the basic Concept of JMCIS and the philosophy behind JMCIS, available on the Internet at http://perch.nosc.mil:8095/Docs/UIS_Supp/

Assumptions.html.

[NRAD02] Navy Research and Development Command and Intelligence Systems Division, Collection of slides that describe the basic concept of JMCIS and the philosophy behind JMCIS, available on the Internet at http://perch.nosc.mil:8095 /Briefs/Briefs.html.

[NRAD03] Navy Research and Development Command and Intelligence Systems Division, Description of the basic concept of JMCIS Common Operating Environment, available on the Internet at http://perch.nosc.mil:8095/Docs/COE/ JMCIS_COE.1.html.

[NRAD04] Navy Research and Development Command and Intelligence Systems Division, Describes of the basic concept of JMCIS Common Operating Environment, available on the Internet at http://perch.nosc.mil:8095/Docs/COE/JMCIS_COE.2.html.

[NSTL94] National Software Testing Lab, *Software Digest: Ratings Report, The Independent Comparative Ratings Report for Selecting PC and LAN Software*, Volume 11, Number 11, November 1994.

[OLSE96] Olsen, Florence, *Microsoft Aims to Fix Up NT for Buyers of POSIX*, Government Computer News, 4 March 1996.

[PABR96] Pabrai, Uday O., Unix Internetworking, Artech House, Boston, 1993.

[PAIG95] Paige, Emmett Jr., *Technical Architecture Framework for Information Management (TAFIM), Version 2.0*, Memorandum from the Office of the Assistant Secretary of Defense, 30 March 1995.

[PARA96] Parameswaran, Ramesh, *Windows Family Interoperability and Integration Within Unix Environments*, Microsoft Corporation, 18 May 1996.

[RAME95] Ramesh, Balasubramaniam, *IS3020 Class Notes*, Naval Postgraduate School, 4th quarter 1995.

[ROYS96] Royster, Curtis Jr., *Maximizing Portability: Achieving Vendor Independence*, , 31 January 1996.

[RULE95] Ruley, John D., Networking Windows NT 3.51, Second Edition, John Wiley & Sons, Inc., 1995.

[RUSS92] Deborah Russell and G.T. Gangemi Sr., Computer Security Basics, O'Reilly & Associates, Inc., Sebastopol, CA., 1992.

[SCHN87] Schneidewind, Norman F., *The State of Software Maintenance*, IEEE, March 1987.

[SCHN96] Schneidewind, Norman F., *Do Standards*, IEEE, January 1996.

[SCHW95] Schwartau, Winn, *One expert shares his views on Microsoft's security features.*, Network World, 23 January 1995.

[SING95] Singh, Inder M., *POSIX has a leading role in open systems*, EE Times Interactive, 18 December 1995.

[SNMP96] Vallil, Tyler, *SNMP VS CMIP: An Introduction to Network Management*, available on the Internet at http://www.undergrad.math.uwaterloo.ca/~tkvallil/snmp.html.

[STEV95] Stevenson, Douglas W., *Network Management: What it is and what it isn't*, available on the Internet at http://smurfland.cit.buffalo.edu/NetMAn/ Doc.Dstevenson, April 1995.

[STRA92] Strassmann, Paul, *Open Systems Implementation and the Technical Reference Model*, Director of Defense Information Memorandum, 12 February 1992.

[TANE92] Tanenbaum, Andrew S., Modern Operating Systems, Prentice Hall, Englewood Cliffs, NJ, 1992.

[UNIF95] UniForum White Paper, *Unix: The Real Success Story*, available on the Internet at http://www.uniforum.org/news/html/publications/UnixWhitePaper.html.

[UWAT96] *CMIP vs. SMIP: An Introduction To Network Management,* available on the Internet at http://www.undergrad.math.uwaterloo.ca/~tkvallil/snmp.html.

[WEBS94] Webster, John, *Spec 1170*, Computer World, 12 September 1994.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   8725 John J. Kingman rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Library, Code 013 — 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Dr. Norman Schneidewind, Code SM/Ss — 1
   Department of Systems Management
   Naval Postgraduate School
   Monterey, California 93943

4. Dr. James Emery, Code 05 — 1
   Associate Provost of Computing Information Systems
   Naval Postgraduate School
   Monterey, California 93943

5. Dr. William E. Ruzicka — 1
   Fleet Numerical Meteorology and Oceanography Center
   7 Grace Hopper Avenue
   Stop 1
   Monterey, California 93943-5501

6. Lieutenant Mark F. Sauer — 1
   2455 Kensington Drive
   Kalamazoo, Michigan 49008

7. Lieutenant Timothy J. Smith — 1
   3590 Syracuse Avenue
   San Diego, California 92122

8. Lieutenant John W. Sprague — 1
   428 Dela Vina Avenue
   Apartment 223
   Monterey, California 93940

9. Lieutenant JG Joseph E. Staier — 1
   2007 Saxon Drive
   New Smyrna Beach, Florida 32169